Object detection with Microsoft HoloLens 2

A comparison between image and point cloud based algorithms

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Geodesy and Geoinformation

by

Sophie Herrmann, BSc
Registration Number 01426553

to the Faculty of Mathematics and Geoinformation

at the TU Wien

Advisor: Univ.Prof. Dr.sc. Ioannis Giannopoulos, MSc BSc
Assistance: Univ.Ass. Dr.phil. Markus Kattenbeck, MA

Vienna, 30th August, 2021

Sophie Herrmann	Ioannis Giannopoulos

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Sophie Herrmann, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 30. August 2021

Sophie Herrmann

Acknowledgements

This thesis would not have been possible without a number of persons. First and foremost I would like to thank the assistant of this thesis, Markus Kattenbeck, who not only allowed me to work on this topic and to adjust it to my interests, but also kept an open mind when discussing questions with me and provided practical ideas for in my opinion complex issues. For me it was extremely helpful that he kept me focused on the general goals of this thesis when I got lost in details.

Next in line have to be my parents, Doris and Gerold Herrmann, who supported me throughout my studies, all financially, morally and with their technical knowledge. They always encourage me to follow and deepen my interests, and never be satisfied with the simple solution.

A special thanks also goes to my boyfriend, Juri Berlanda, who supported me both with technological insights and morally throughout the becoming of this thesis. This combination made him my illuminating benefit, especially during difficult times.

I would like to also thank the advisor of this thesis Prof. Ioannis Giannopoulos, who was an invaluable source of knowledge and experience especially in the object detection domain. He provided me useful suggestions for my initial design and consecutive improvements which continuously guided me into the right direction.

Finally, I would like to thank Bartosz Mazurkiewicz and Negar Alinaghi two assistants of the Research Unit of Geoinformation, who supported me with the realization of my data acquisition experiment. This includes both the technological implementation and, foremost, the physical setup, which would not have been possible without them.

Abstract

Object detection is a central task in computer vision, which tries to identify and locate objects in a scene. After the era of hand-crafted features and limited accuracy, 2014 AlexNet brought the attention to neural networks and deep learning. Accuracy increased and inference time decreased significantly and it came to a boom of 2D image-based object detection networks. Nowadays a huge number of 2D object detection algorithms are available, but they have one main drawback, they only provide two dimensional bounding boxes. This is an issue as a number of modern applications operate in the three dimensional world, therefore also require accurate 3D information. Mainly over the last decade 3D object detection algorithms evolved leveraging deep neural networks. This thesis focuses on comparing accuracy and inference time of 2D and 3D object detection algorithms, evaluating whether the additional information provided by 3D algorithms comes at the cost of lower accuracy and / or slower inference times.

To enable such an evaluation a comparable 2D - 3D dataset is created. Due to the availability of both a 2D and 3D measurement device and ease of use, head mounted Augmented Reality glasses are used for data acquisition. Special considerations are applied to the data acquisition setup, selected categories and environmental / lightning conditions, to allow detailed analysis of key properties of 2D respectively 3D object detection algorithms. Acquired data is further processed to clean data and annotate objects. The comparison is based on one representative object detection algorithm per domain, YOLOv3 is selected as 2D algorithm, VoteNet as 3D algorithm. Multiple sets of hyperparameters are tested and the best models are finally compared.

The evaluation shows that 2D and 3D accuracy results are with $AP_{2D-50} = 0.94$ and $AP_{3D-50} = 0.96$ very similar and 3D results are even slightly better. The main downside of 3D compared to 2D algorithms is the inference time which is — though achieving real-time — with 19.04ms respectively 1.55ms by a factor ten slower in this experiment. A detailed analysis per category and condition shows that the accuracy of the 3D algorithm mainly depends on the number of object points and their density, while 2D algorithms benefit most from large object size.

Contents

List of Figures

List of Tables

Listings

CHAPTER 1

Introduction

1.1 Motivation

A central goal in computer vision is to enable computers to understand scenes [Aziz et al., 2020]. Object detection is an important piece to reach this goal, as it tries to answer the question

What objects are where?

Compared to classification it adds additional information about the location of each object instance. This central computer vision task is the basis for a number of other computer vision tasks such as image captioning (see e.g. [Karpathy and Fei-Fei, 2015]), object tracking (see e.g.[Kang et al., 2017]), and instance segmentation (see e.g. [Hariharan et al., 2014]). Until recent years the main focus of object detection was on images — 2D data.

To solve the stated question computational models and techniques are developed. Before the era of deep neural networks 2D object detection was based on hand crafted features such as scale-invariant feature transform (SIFT) [Lowe, 1999] and histogram of oriented gradients (HOG) [Dalal and Triggs, 2005] which achieved only limited accuracy. The modeling strategy changed completely when [Krizhevsky et al., 2012] presented AlexNet in 2014 — the first neural network for object classification. Shortly after [Girshick et al., 2014] was able to leverage neural networks not only to classify an image by its content but also locate specific objects within the image, thereby performing object detection. From roughly this point onward object detection with deep neural networks developed fast. Hundreds of networks got developed continuously improving the accuracy and speed of object detection. Today small and fast networks already achieve reasonable accuracy and more complex networks provide even more precise results [Xiao et al., 2020] (see Subsection 2.1.6). The growing interest is also reflected in the increasing number of publication in the field of object detection. According to the systematic literature review

by [Zou et al., 2019] the number of publications increased from roughly 20 in 1998, over a bit less then 400 in 2008, to nearly 1200 in 2018. This enormous interest of academia also facilitated a wide range of real-world application such as autonomous driving, robot vision and video surveillance.

Hence, this boom of 2D object detection algorithms can be mainly attributed to the usage of deep neuronal networks. The rapid development is additionally enabled by the increasing availability of computational resources. Many modern deep neural networks leverage but also require access to fast processing units and a sufficient amount of memory [Strubell et al., 2019].

As described above, nowadays 2D object detection is used by a number of applications. But in recent years for more and more applications 2D information is not sufficient. Instead accurate 3D locations are required. One central reason for this demand is that some modern applications — such as autonomously driving cars, automated production lanes and Augmented Reality (AR) — operated in the three dimensional world. An AR application requires 3D information to properly place virtual content onto real objects, or to provide meaningful 3D distance measures for navigation. Motivated by the good results of 2D object detection and enabled by availability of large scale computational resources, modern 3D object detection algorithms are often based on deep neural networks. Although most development happened only throughout the last decade already good accuracy can be achieved today [Rahman et al., 2019].

Compared to the 2D domain 3D object detection has a number of advantages, most importantly the additional information about the third dimension. But this additional information also requires additional considerations. The most practical question examines available sensing devices. 3D data can be acquired in different way, e.g. with laser scanners, via 3D point reconstruction from a least two images or a combination of multiple tools (e.g. used by modern AR devices). Connected to the sensing device is the data structure, 3D data can be stored in different formats such as point clouds, voxels (3D pixels), or 2D projections. In most cases 3D data requires more storage and processing resources because of the additional dimension. Each of the described choices provides different advantages and disadvantages (see Subsection 2.2.3). Therefore up to now no standard choices to these options has been defined [Shen, 2019]. Being such a young and diverse research field makes 3D object detection especially interesting.

3D object detection algorithms evolved because the results of accurate 2D algorithms are missing required information about the third dimension. With this group of algorithms 3D results can be provided. Still, 3D object detection is a quite young research area developing into different direction driven by a wide range of applications. Several questions arise. How good are 3D object detection algorithms compared to their 2D equivalents? Do we get the additional information about the third dimension at the cost of a lower accuracy and slower detection, or can 3D algorithms compete with 2D ones?

As stated above Augmented Reality (AR) is one of the applications which would at least strongly benefit from 3D object detection as it operates in the three dimensional world.

Generally, AR is able to combine the real world with virtual content (see Subsection 2.3.1). A typical setup is a head mounted device which visualize virtual objects via AR glasses. Applications range from interactive games over manufacturing purposes to simplified learning. Interestingly, to the author's best knowledge, currently only 2D object detection is used with AR (see e.g. [Lee et al., 2019] and [Mahurkar, 2018]). This is a waste of resources as many AR systems capture and provide both 2D and 3D information of their surrounding environment. Those two properties make AR a very interesting device to realize the aforementioned comparison.

1.2 Goals

As motivated above, the central goal of this thesis is to compare 2D and 3D object detection leveraging AR. The basis for such a comparison is a comparable 2D - 3D dataset. Such a dataset is especially important for the comparison between state-of-the-art object detection algorithms because most of them leverage neural networks [Zou et al., 2019], [Rahman et al., 2019] which strongly depend on the provided training data [Bengio et al., 2017, p. 98-100]. Up to now no such dataset exist (see Subsection 2.1.2 and 2.2.2). Hence, this is the first fine-grain target of this thesis: generating such a dataset.

Although no AR generated dataset exists, AR allows to easily acquired such a dataset — AR is even seen as a simple mapping device by some applications (e.g. [Hübner et al., 2020], [Khoshelham et al., 2019]). But, the availability of 2D and 3D data is not the only advantages of AR devices. As head mounted device it provides useful flexibility. Hardly any setup is required before starting an observation. AR can easily be used in different environment, e.g. both in indoor and outdoor, observing objects of varying size. This allows to easily include acquisitions from multiple environments into the dataset. In summary, AR is a good choice for creating a comparable dataset.

Based on this new AR dataset 2D and 3D algorithms are evaluated. The focus is on an accuracy and inference time evaluation. Such an analysis of object detection results in addition to the availability of a AR dataset supports (3D) object detection developments in this emerging AR area. The analysis of accuracy results provides insights into the applicability of AR data to certain object detection algorithms. And, the comparison of inference times shows whether and how AR and object detection can be combined for certain applications.

The focus on the AR domain should further show that autonomous driving is not the only application of 3D object detection. Rather a number of other applications already exist which could — and maybe even should — consider further development into this direction. The AR-object detection could even be seen as a prototype for other applications concentrating on this combination.

Next to support efforts to bring 3D object detection to new applications the focus of this thesis is to relate 2D and 3D object detection. Two topics are considered in this regard: First, the overall accuracy of 2D and 3D object detection should be compared. Second,

strength and weaknesses should be analyzed in detail. 2D and 3D data may depict the same scenario but still provide very different information. Because of different properties of data from the two domains it is expected that they also perform better respectively worse on certain categories and / or conditions. The goal of this thesis is to identity key properties which improve or worsen detection accuracy. In a next step also synergies of the domains are of interest.

1.3 Overview

This thesis is structured as following: Chapter 2 presents a literature review of the three important components of this thesis, 2D object detection (Section 2.1), 3D object detection (Section 2.2), and Augmented Reality (Section 2.3). The first starts with an general overview and definition of 2D object detection and positions object detection in relation to other computer vision tasks. Then, publicly available datasets and corresponding evaluation metrics are presented, especially focusing on widely used accuracy metrics and their variations. After a short description of the origins of 2D object detection and the era of hand crafted features, deep learning and convolutional neural networks are presented in detail. Following this theoretically explanation the most important convoluational neural networks and their performance are discussed. The 3D object detection section is structured similarly. It starts with an overview about 3D object detection, including a motivation for the additional dimension and connected considerations. Then, again datasets and metrics are discussed, focusing also on the differences and similarities between the 2D and 3D domain. Important 3D algorithms with their different focuses are presented and subsequently their accuracy and detection speed are compared. The last section defines Augmented Reality, connects AR with object detection by presenting a number of interesting AR applications and finally describes properties of the used AR device — Microsoft HoloLens 2. The final Section 2.4 details the contributions of this thesis based on the provided literature review.

The subsequent Chapter 3 explains the acquired dataset. It is divided into three parts, Methodology (Section 3.1), Results (Section 3.2) and Evaluation (Section 3.3). The first section details the creation of the new 2D-3D dataset. Firstly, dataset requirements are stated, mainly focusing on category properties, environmental condition, and the measurement device. Secondly, a dataset concept is presented fulfilling the before described requirements. Thirdly, the technical setup is detailed. Selected technology is described and implemented software is presented. Fourthly, the acquisition configuration is given. The acquisition process — including physical setup and selected parameters — is described. Finally, a description of required processing of the 2D respectively 3D data is given. This includes outlier removal and annotation. In the second section results of this process are given. First, observations made during the acquisitions are summarized, resulting images and point clouds are visualized, and key information about the acquired data is stated. Then, processing results for 2D and 3D data given separately. The 2D part mainly focuses on the results of the transfer learning for the 2D labeling process, including accuracy measures and sample images. The 3D part present the results

of the algorithmic bounding box derivation and volume analysis. In the final section characteristics of the new dataset are evaluated. It relates the 2D and 3D part of the dataset of existing standard datasets from the respective domain, discusses results of the data acquisition setup and analyzes implications of the measurement device HoloLens 2 on the different categories and conditions.

Following the dataset description, Chapter 4 presented the applied object detection analysis. It focuses on the selection and implementation of the object detection algorithms. To compare 2D and 3D object detection, one algorithm each should be selected. Therefore, the chapter is divided into three parts, Algorithm Requirements (Section 4.1), Algorithm Selection and Training (Section 4.2), and Comparison Strategy (Section 4.3). Based in the requirements stated in the first section, an algorithm is selected in the latter section. Algorithm and thraining is described. The chapter closes with a description of the used comparison strategies, focusing in accuracy and detection speed.

After presenting the method, the results of the experiment are given in Chapter 5. It summarizes the results of the main object detection experiment. Accuracy values and inference times for both 2D and 3D domain are shown.

The results are evaluated in the following Chapter 6. First, results of the object detection experiment are positioned in relation to previous research (Section 6.1), and then 2D and 3D results are compared (Section 6.2). The discussion of object detection results per domain focuses — similarly as for the dataset — on positioning and comparing results to previous research. Additionally, category and condition specific variation are discussed. For the 2D domain also the effect of data augmentation is discussed. The final comparison evaluated the accuracy and inference time of the algorithms and describes implications of the results.

In the final Conclusion and Future Work Chapter 7 first the methodology, results and evaluation are summarized and most important findings are detailed. Then, remaining and resulting open questions are stated and possible strategies are described how they could be further analyzed.

CHAPTER 2

Related Work

In this section a literature review of the evolution and current state of object detection is given. The following sections cover different aspects of object detection especially focusing on different input data. First, object detection based on 2D data is presented, then the evolution of 3D data based methods is covered, and in the last section the connection to Augmented Reality is drawn.

2.1 2D Object Detection

2.1.1 Overview

In computer vision an important goals is understanding images [Aziz et al., 2020]. To reach this goal one must capture the content of an image. This can be done in different ways resulting in four main computer vision tasks as show in Figure 2.1. The simplest task is *image classification* which identifies important objects in an image and outputs the relevant categories. The second task *object detection* not only outputs the relevant categories but also the location of the detected objects as axis-aligned rectangular bounding boxes. In other words it combines the tasks of object classification and object localization. The other two options to retrieve the location of an object is to perform *semantic segmentation* or *instance segmentation*. Both tasks output the relevant class on a pixel level. The difference between these tasks is that semantic segmentation does not differentiate between instances of a single category whereas instance segmentation separates multiple instances of the same category [Sultana et al., 2019].

Next to the above described classification of computer vision task also others exist. Particularly important for this thesis is *object recognition* which can have a similar meaning as *object detection* [Andreopoulos and Tsotsos, 2013]. For clarity it should be stated that in this thesis only the wording *object detection* will be used with the meaning as defined in [Zhao et al., 2019b, p. 3214]:

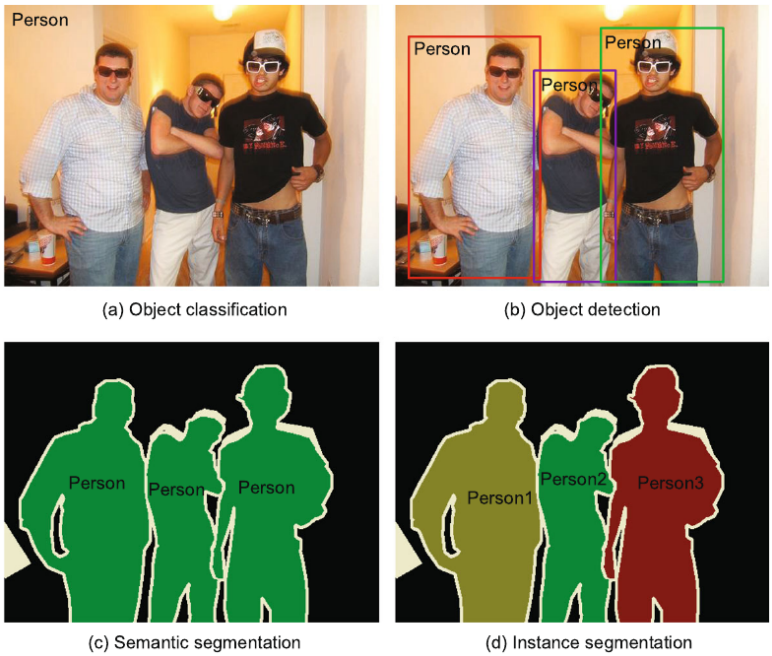


Figure 2.1: Four main computer vision tasks, (a) *object classification* states that there is an object in the image, (b) *object detection* classifies and locates objects in an image, (c) *semantic segmentation* classifies objects on a per pixel level, and (d) *instance segmentation* also returns classification on a per pixel level with additional distinction between separate instances of the same category. Figure taken from [Xiao et al., 2020, p. 23730].

[...] object detection aims at locating and classifying existing objects in any one image and labeling them with rectangular BBs [bounding boxes] to show the confidences of existence.

Object detection is further separated into *generic* and *dedicated* object detection. The first one summarizes general methods to detect very different object categories. In contrast, the second one focuses on one important object category. Examples are face-, pedestrian- and vehicle detection [Xiao et al., 2020], [Zhao et al., 2019b]. In this thesis the focus will be on generic object detection.

Independent of the type of object detection one central step is to return bounding boxes of objects. In 2D object detection bounding boxes are axis aligned and are therefore unambiguously defined by four parameters. Some representation formats are more common than others [Sultana et al., 2019]. For instance, a bounding box can be defined by its upper-left coordinates (x, y) and its absolute *width* and *height* in pixels or by its relative coordinates normalized to the image size $(x_{rel}, y_{rel}, width_{rel}, height_{rel})$. The most common representation formats today were introduced either by labeling software or datasets [Padilla et al., 2021].

2.1.2 Datasets and Metrics

Over the last decades many different object detection algorithms evolved both in the area of generic and dedicated object detection. To compare the performance of new methods a set of standard datasets together with evaluation methods emerged

Datasets

Next to a number of new object detection algorithms the last 20 years brought up a number of annotated object detection datasets. Due to the great performance of some datasets they became important indicators to measure the performance of algorithms. Using unified input data differences between local datasets can be eliminated and a comparable evaluation is possible [Xiao et al., 2020].

The size of a dataset can range from some MB to TB of data [Xiao et al., 2020]. To create such large datasets crowd funding strategies are often applied. The general procedure of generating a dataset can be summarized as following. First, the object categories are defined, second, a set of images showing divers instances of the defined categories are collected and third, the images are annotated [Aziz et al., 2020]. Finally, the dataset is split into train, validation, and test set.

Most datasets are published within a particular object detection challenge. In the field of generic object detection five datasets are very popular [Zou et al., 2019]: PASCAL VOC2007 [Everingham et al., 2007], PASCAL VOC2012 [Everingham et al., 2012], ImageNet [Deng et al., 2009], Microsoft COCO [Lin et al., 2014], and OpenImages [Krasin et al., 2017]. Their statistics are summarized in Table 2.1. Next to the above mentioned generic object detection datasets a number of dedicated object detection datasets exists focusing for instance on pedestrian or face detection [Sharma and Mir, 2020].

Dataset	# Classes	train		validation		test	
		images	objects	images	objects	images	objects
VOC-2007	20	2,501	6,301	2,510	6,307	4,652	14,976
VOC-2012	20	5,717	13,609	5,823	13,842	10,991	-
ILSVRC-2017	200	456,567	478,807	20,121	55,502	65,500	-
MS-COCO-2018	80	118,287	860,001	5,000	36,781	40,670	-
OID-2018	600	1,743,042	14,610,229	41,620	204,621	125,436	625,282

Table 2.1: Five most popular 2D generic object detection datasets with their statistics. Table adopted from [Zou et al., 2019, p. 6].

PASCAL VOC (Visual Object Classes) Challenge[1]. The pioneer in the field of algorithm competition, publishing large-scale comparable data, was PASCAL VOC. The challenge was held from 2005 to 2012 each year. The most popular challenges and

[1] http://host.robots.ox.ac.uk/pascal/VOC/, accessed 9-April-2021

Figure 2.2: Difference between iconic (left two) and non-iconic objects (right two). It is argued that the later are more difficult to detect. (Images taken from [Lin et al., 2014].)

datasets are the ones from VOC-2007 and VOC-2012. The first one contains 5k images respectively 12k annotated objects and the second one 11k images with 27k objects. 20 categories of commodity items are annotated in both datasets [Everingham et al., 2007], [Everingham et al., 2012]. Although Pascal VOC was a popular dataset at its time, nowadays its popularity decreases in favor of improved datasets like MS COCO and it is rather used as a test bed for new algorithms [Zou et al., 2019].

ILSVRC (ImageNet Large Scale Visual Recognition Challenge)[2]: This challenge was held from 2010 to 2017. It drastically increased the number of categories, images and annotated objects, in detail the number of categories was increased by a factor of 10 to 200 categories and the number of images and annotated objects is two orders of magnitude bigger with about 517k images and 534k annotated objects 2017 [Deng et al., 2009], [Zou et al., 2019].

MS COCO (Microsoft COCO)[3]: The currently most challenging dataset is MS COCO. The corresponding annual competition has been held since 2015. Compared to its predecessor ImageNet the number of categories is reduced to from 200 to 80. In contrast, the number of annotated objects strongly increased. The goal of MS COCO is to better cover real world examples. Therefore the focus is on non-iconic images (see Figure 2.2). This includes objects which are amid clutter or heavily occluded. Non-iconic also means that a wide range of object scales are covered with a strong focus on small objects (where the object covers less then 1% of the image). Including all these challenges MS COCO currently is the de facto standard for evaluating generic object detection algorithms [Lin et al., 2014], [Zou et al., 2019].

OpenImages / Open Images Detection (OID) challenge[4]: The newest dataset in this list is OpenImages. It was introduced in 2018 and is currently the largest publicly available dataset with 1,910k images and 15,440k annotated objects from 600 categories [Krasin et al., 2017].

[2] http://image-net.org/challenges/LSVRC/, accessed 9-April-2021
[3] http://cocodataset.org/, accessed 9-April-2021
[4] https://storage.googleapis.com/openimages/web/index.html, accessed 9-April-2021

Metrics

As mentioned above, standard datasets ensure comparability between algorithms. Comparison is performed based on algorithms performance measured by a set of evaluation criteria. In general two main groups of performance measures can be considered, first, detection speed and second, the accuracy of the predicted result.

The speed of a detection can either be measured as inference time per image in milliseconds or as processable frames per second (fps) in Hertz. The detection speed is important for real-time application, where real-time is defined as a frame rate of 30 or higher or by an inference time smaller than 33.3 ms [Redmon et al., 2016].

In object detection accuracy can be evaluated matching a given ground-truth bounding box B_{gt} to the output of a detection system: a predicted bounding box B_p, its predicted category and its confidence. First a confidence threshold β must be set to define which detections are considered [Liu et al., 2020]. To measure the geometrical similarity between a predicted and ground-truth bounding box of the same category *Intersection over Union (IoU)* can be used [Xiao et al., 2020]. According to [Everingham et al., 2010, p. 314] IoU is defined as following:

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} = \frac{area \quad of \quad overlap}{area \quad of \quad union} \tag{2.1}$$

The resulting coefficient is between 0 and 1, where higher values indicate a better match. Object detection metrics use different IoU thresholds ε to identify a correct detection. The higher the threshold the more restrictive the measure is. With a higher threshold a predicted bounding box must better match the ground truth to be considered a correct detection. Therefore a comparison between different thresholds reveals the quality of predicted bounding boxes [Padilla et al., 2021].

IoU is used to calculate *Precision (P)* and *Recall (R)* per category C_i and instance j [Xiao et al., 2020, p. 23768]. Precision measure whether only relevant object are detected. Recall measure whether all relevant objects are detected. These metrics are defined using basic statistical concepts: True Positive (TP - correct detection), False Positive (FP - incorrect detection) and False Negative (FN - missing detection). True Negatives are meaningless in the field of object detection as there is an infinite number of correctly not detected bounding boxes [Padilla et al., 2021].

$$P_{C_{ij}} = \frac{TP_{C_{ij}}}{TP_{C_{ij}} + FP_{C_{ij}}} \qquad R_{C_{ij}} = \frac{TP_{C_{ij}}}{TP_{C_{ij}} + FN_{C_{ij}}} \tag{2.2}$$

One precision value per category is calculated as the *Average Precision (AP_{2D})*. The suffix $_{2D}$ is used to highlight that an accuracy describe 2D metric. This is done to distinguish between 2D and 3D metrics introduced in following sections. As described in

[Padilla et al., 2021] different implementations are available. Here the general equation for averaging over all instances is given as presented in [Xiao et al., 2020, p. 23768].

$$AP_{C_i} = \frac{1}{m} \sum_{j=1}^{m} P_{C_{ij}} \tag{2.3}$$

To then get an average over all categories the *mean Average Precision (*mAP_{2D}*)* can be used [Xiao et al., 2020, p. 23768]:

$$mAP = \frac{1}{n} \sum_{i=1}^{N} AP_{C_i} \tag{2.4}$$

Based on IoU threshold, AP_{2D}, and mAP_{2D} a number of variations were introduced. MS COCO directs the focus towards improving the location of detected bounding boxes. This is realized by not only evaluating AP_{2D} for an IoU of 0.5 as in PASCAL VOC, but by using different threshold values for IoU, AP@.5 (AP_{2D-50}), AP@.75 (AP_{2D-75}) and AP@[.5:.05:.95] are employed. The first two provide AP for a threshold of 0.5 respectively 0.75. The latter averages the AP for 10 IoU thresholds [0.5, 0.55, 0.6, ... 0.95] into a single new value [Lin et al., 2014]. To consider the effect of object size *AP Across Scale* can be used. Mean Average Precision for small (AP_{2D-S}), medium (AP_{2D-M}), and large (AP_{2D-L}) objects are calculated separately [Lin et al., 2014]. Equivalent to the Average Precision also the Average Recall can be calculated replacing P with R in Equation 2.3. AR_{2D} can be calculated separated by the number of detection per image. Often used values for the maximum number of detection are 1 (AR_{2D-1}), 10 (AR_{2D-10}) and 100 (AR_{2D-100}). Additionally variations based on object scale (AR_{2D-S}, AR_{2D-M}, AR_{2D-L} - *AR Across Scale*) and IoU are also possible [Padilla et al., 2021]. All of the above mentioned measures depend on the selected confidence threshold (β) determining the set of accepted bounding boxes [Liu et al., 2020].

2.1.3 History of object detection

The initial attempts to recognize simple patterns in images can be dated back to the 1950s and 1960s. Initially the focus was on geometric representations and pattern matching [Mundy, 2006]. [Attneave and Arnoult, 1956] were the first to analyzed the human perception of primitive shapes and patterns. [Roberts, 1963] described 3D reconstruction and identification of primitive shapes and convex polygons based on contours. Leveraging the idea of contour based detection, edge filtering became an important tool for early object detection. [Harris et al., 1988] successfully constructed a 3D description of a more complex shape, an airplane. Such descriptions are required as templates to match and detect objects. [Fischler and Elschlager, 1973] detailed a template matching technique based on 2D geometric features.

In the 1990s the focus shifted towards statistical classifiers based on appearance features [Mundy, 2006]. [Murase and Nayar, 1995] describes objects as their appearance manifolds

leveraging and combining shape, reflectance, pose and illumination. [Schmid and Mohr, 1997] present local invariant intensity features for object retrieval.

In this era of traditional object detection the detection process can be separated into three main steps: first, region selection or keypoint detection, second, feature extraction and third, classification. In the first step the whole image is scanned to find regions with potential objects. This is required as different objects may appear at different locations, with different aspect ration or different size. In the second step a descriptive and robust representation is extracted. And, in the third step a classifier distinguished between the object categories based on the feature description [Zhao et al., 2019b].

For feature extraction a wide range of handcrafted feature representations and classifiers were developed. Feature vectors should be robust in the sense of being invariant to a number of variations like translation, scale, rotation, illumination, background and occlusion [Liu et al., 2020], [Zhao et al., 2019b]. An important example for a robust feature vector is the so-called scale-invariant feature transform (SIFT). SIFT combines multiple scales, using the difference of Gaussians and histogram of gradients [Lowe, 1999], [Lowe, 2004]. The histogram of oriented gradients (HOG) detector is a well-known advancement of SIFT enabling the user to detect objects of different size [Dalal and Triggs, 2005]. Due to the limited computational resources at the time also the development of speed up skills was a crucial part. The Viola Jones detectors were the first ones to achieve a detection speed of 15 fps [Viola and Jones, 2001], [Viola and Jones, 2004].

Between 2010 and 2012 the object detection performance — based on hand-crafted features — plateaued. Only minor improvements could be achieved by combining existing systems or proposing variations of successful techniques [Zou et al., 2019]. The ImageNet Challenge 2012 saw the rebirth of convolutional neural networks (CNNs) with the winning AlexNet. Image classification error rates could be halved [Krizhevsky et al., 2012]. This showed that deep neural networks (DNN) are capable of learning robust features. Finally 2014 Region with CNN implemented not only object classification but object detection with CNNs [Girshick et al., 2014], [Girshick et al., 2015]. This was the starting point for an era of deep learning and CNN based object detection.

The following subsection gives more details about this game changing idea of machine learning and (convolutional) neural networks.

2.1.4 Deep learning and Convolutional Neural Networks

In recent years, *machine learning* has become an omnipresent technology with a range of applications in science, business, and government, alike. Examples include object classification and detection [Krizhevsky et al., 2012], [Szegedy et al., 2015], speech recognition [Hinton et al., 2012], [Sainath et al., 2013], and medical analysis [Ma et al., 2015], [Helmstaedter et al., 2013]. Machine learning both powers web search engines and can be found in consumer products like smartphones [LeCun et al., 2015].

One of the most important concepts behind it is *supervised learning*. Both input and expected output are given and the application learns the relation or mapping between

those. This is achieved by finding and learning interesting patterns in the data, e.g. reoccurring geometries, intensity distributions or combinations of color features. In other words most machine learning algorithms are nowadays learning from data [Bengio et al., 2017, p. 98-100].

Today more and more data is accessible, including large-scale annotated datasets. Previous machine learning algorithms could not benefit from this ever growing supply of data. Therefore bigger and more complex algorithms were required, establishing the field of *deep learning*. No unique definition of deep learning exists but the common idea is to learn feature representations in a hierarchical manner, exploiting the idea that higher-level feature can be composed of lower-level features [Zhang et al., 2018].

One central property of the hierarchical mapping is non-linearity [Deng and Yu, 2014]. Only a non-linear mapping can be a robust feature descriptor, meaning that the mapping can be sensitive to small details in e.g. geometry while being insensitive to large irrelevant variations such as background, scale or illumination. The non-linearity distorts the feature space from lower-level to higher-level features to make the target features linearly separable at the highest level [LeCun et al., 2015].

The advantage of deep learning compared to hand-crafted features is that feature descriptors are learned automatically from data. Before, a good data — and specifically image — representation was missing. Therefore sophisticated feature representations were developed as presented in Section 2.1.3. With deep learning this is not required anymore. Applications find the best data and feature representation by themselves, removing the limiting dependency on manual human interaction and domain specific expert knowledge in the field of feature extraction [Zou et al., 2019].

In machine / deep learning a wide range of algorithms can be used to model data and find the best mapping between input and output. A popular choice are *neural networks* which model data as graph of *neurons* or *units*. It can be summarized as multi-layer stack of so-called *layers* grouping together a set of units. Generally, a neural network is composed of an input layer, a set of hidden layers and an output layer [LeCun et al., 2015]. Figure 2.3 shows a very simple neural network. *Deep neural networks* employ a high number of hidden layers [Bengio, 2009, p. 16].

In the field of neural networks the learning process is called *training*. It requires multiple steps. First, in the *forward propagation* the network gets input data and predicts an output. Then an objective function — called *cost* or *loss function* — is calculated to compare the predicted with the true output. During *backward propagation* the network parameters are adjusted to reduce the distance between true and predicted output and therefore optimize the cost function [LeCun et al., 2015].

In detail, network parameters are floating point values which defined the relation between consecutive layers and their units as presented in Figure 2.3. The input to a unit y_j is calculated as the weighted sum of all units in the previous layer plus a bias (Equation 2.5) to which a non-linear function is applied (Equation 2.6) [Liu et al., 2020, p. 268]. In the first part the trainable parameters — weights and bias — are applied. Each unit

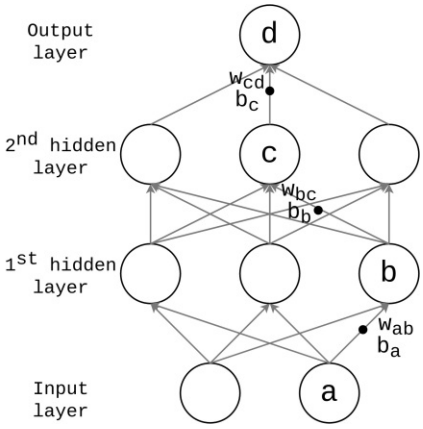


Figure 2.3: The basic structure of a neural network with an input, two hidden and an output layer. The arrows symbolize the relation between the units in a forward pass. To derive an unit in one layer from its predecessors, weights and bias are needed (Equation 2.5). Inverting the arrows would show the direction and relations during backward propagation. (Based on [LeCun et al., 2015, p. 437])

has its own weight vector and bias. The second part ensures a non-linear input - output mapping [Liu et al., 2020].

$$z_j = \sum_{i=1}^{n} w_{ij} x_i + b_i \tag{2.5}$$

$$y_j = \sigma(z_j) \tag{2.6}$$

The non-linear function is often called *activation function* as its output is the input or activation of another unit. Currently the most popular activation function σ is the Rectified Linear Unit (RELU) (Equation 2.7) [LeCun et al., 2015]. Compared to other, previously used alternatives — like sigmoid or hyperbolic tangent — it can be calculated faster and therefore increases training speed significantly [Glorot et al., 2011].

$$\sigma(z) = max(0, z) \tag{2.7}$$

The whole optimization task of training parameters is nowadays mostly implemented as Stochastic Gradient Descent (SGD). The error between predicted and true output is reduced by descending along the gradient of the cost function (e.g. cross entropy loss or mean squared error [Xiao et al., 2020]). SGD calculates the required weight changes by applying the chain rule of derivatives from the output to the input layer and propagates the gradients through the layers. Inverting the arrows depicted in Figure 2.3 shows the relation between the single units in backpropagation. Compared to other, more elaborate optimization strategies the simple SGD finds a good set of network parameters quickly [Bottou and Bousquet, 2011].

Before updating weights and bias a fixed number of input - output pairs are computed. This number is called *batch size*. A batch size smaller than the number of all examples may be required due to computational resource limitations. Furthermore, progress on parameters optimization is made faster with a smaller batch size. Still a too small batch size may slow down training as available storage and processing resources may not be used exhaustively and software optimization and acceleration techniques can only exploit their full potential for large batches. Hence, it is important to find a meaningful value between one and all examples [Peng et al., 2018]. When the network has seen all examples once, also if they were in different batches, one so-called *epoch* of training has been completed [Breuel, 2015].

Different improvements of SGD have been developed. One enhancement is Adaptive moment estimation (Adam). Compared to other extensions empirical tests showed that it works well for many different algorithms [Dogo et al., 2018]. It makes the learning more robust and can increase learning speed [Kingma and Ba, 2014]. SGD and Adam must be parameterized. Both depend on the so-called *learning rate* α which determines the step size in the direction of the steepest gradient descent. Too big values may overshoot the optimal solution, while too small values lead to slow training [Breuel, 2015].

The learning rate and batch size are examples of *hyperparameters*. Compared to the network parameters weights and bias, hyperparameters are not learned during training but must be defined before training. Hyperparameters describe the network architecture and parameterize the training process. A non-extensive list includes: number of hidden layers, number of units per layer, used activation function, learning rate, additional parameters for Adam, batch size and number of epochs [Breuel, 2015]. Currently no automatic procedure exits to obtain an optimal set of hyperparameters for a given network. Hence, finding a good set of hyperparameters requires empirical testing of different hyperparameter combinations [Claesen and De Moor, 2015].

When optimizing hyperparameters like number of hidden layers and number of units one also has to consider the effects on the trainable parameters. Due to the higher number of trainable parameters in neural networks the issue of overfitting may arise. Overfitting means that the network not only learns the characteristics of the target but also random noise of the training samples. This reduces the effectiveness of the network on new samples. The network searches in new samples not only for the target feature but also for the surrounding noise which does not match. Different regularizations strategies have been introduced to reduce the effect of overfitting. One option is to extend the cost function to not only measure the distance between predicted and true output but also to ensure that weights are kept small. This is called L_1 or L_2 regularization depending on the used metric in the cost function [Kukačka et al., 2017]. Another widely used strategy is dropout which randomly removes single units during training. The idea is to make every single unit replaceable and no higher layer units solely depend on a small number of lower layer units with extremely high weights [Srivastava et al., 2014]. A third option is to ensure that the network sees enough different sample during training so it does not learn random noise. This can be achieved by either creating a significant amount of new

training samples or altering existing training samples. The first may be expensive and not always feasible, while the second — also called data augmentation — is usually a cheap and simple process. Examples of data augmentation in the field of image processing are flipping, cropping, random noise injection and color space transformations [Shorten and Khoshgoftaar, 2019].

In some domains it can be challenging to create even a large annotated dataset to get meaningful results from data augmentation. In e.g. bioinformatics or robotics data acquisition and data annotation can be quite costly or time consuming. To overcome this problem of too small training datasets transfer learning can be used. The idea is to transfer or extent knowledge from one problem to another. In practice this means using a model already pre-trained on e.g. a standard dataset such as MS COCO as starting point for another domain specific training. According to the concept of hierarchical feature representations lower level feature are useful for a wide range of higher level feature. Depending on the size of the domain specific dataset only the last layer's parameters or even all parameters may be improved during the domain specific training [Tan et al., 2018].

The concepts presented so far describe the architecture and functionality of general neural networks which expects an one-dimensional input vector. Covolutional neural networks are designed to operate on multi-dimensional arrays such as a 2D image with three color channels, called feature maps [LeCun et al., 2015]. The basic building blocks of CNNs are convolutional (conv.), pooling (pool.) and fully connected (FC) layers. Weights in a conv. layer are grouped together into filter kernels which are used by each unit of the layer. A filter kernel defines for each unit a local patch of units in the previous layer which are used for its computation. More precisely, a unit is calculated as a local weighted sum plus bias to which an activation function is applied. Compared to Equation 2.5 and 2.6 not all units in the previous layer are used and the same weights are applied to all units of the current layer. Multiple m filter kernels can be used in a single layer resulting in m feature maps in the next layer [Liu et al., 2020]. In other words the number of feature maps can be adopted by defining a correct number of filter kernels. To change the size of a feature map pooling layers are used. Pooling merges multiple features into one. The most popular type of pooling is max pooling which keeps only the highest and supposedly most important feature in a neighborhood. This way the dimension can be reduced but the relative location between features are kept. Figure 2.4 shows an often applied structure in CNNs. First one or multiple conv. layers followed by a max pool. layer. A network is often composed of multiple of these blocks which continuously increase the number of feature maps while reducing the size of the feature maps. In the end some FC layers are applied. FC layers are standard neural network layers as described before which are used for fine-tuning in CNNs. Due to their structure FC layers are responsible for a significant amount of weights [LeCun et al., 2015].

As CNNs only extend the principals of standard neural networks they can similarly be trained with forward and backward propagation applying SGD. Also, the same strategies to reduce overfitting and similar hyperparameters need to be selected, but in CNNs also

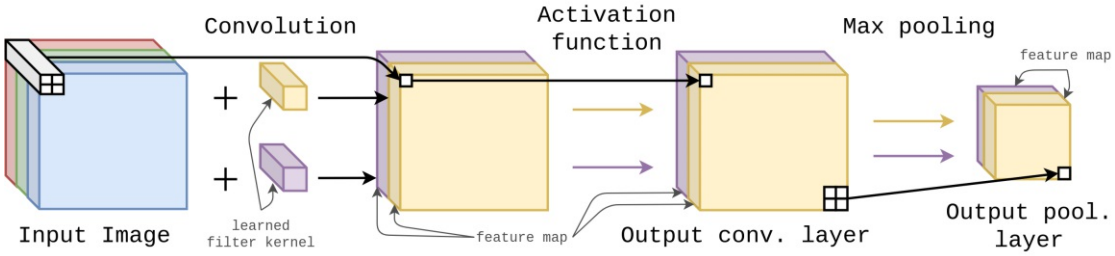


Figure 2.4: A typical group of the two most important layer types in CNNs. First a conv. layer with two filter kernels is applied to a color image. This results in two feature maps. Then a pooling layer follows reducing the size of the feature maps. Concept taken from [Bengio et al., 2017, p. 305]

the number of filter kernels and their properties need to be defined [Bengio et al., 2017, p. 328f].

Deep convolutional neural networks have important advantages especially for the task of object detection. It applies the idea of a hierarchical composition of features. Filter kernels in the first layers could extract edges. Then those edges are combined to simple geometries by higher level filter kernels, which are then in turn combined to even more complex motifs like faces. This example directly shows another effect, deeper networks provide higher expressive capabilities [Zhao et al., 2019b]. One major difference between standard and convolutional neural networks is the idea of persisting local connections. The probability that geometrically close pixels are related is higher then for pixels which are further apart. CNNs keep and use this information. Additionally, filter kernels — as shared weights — realize the concept of location invariance. Simplified, a filter kernel which detects faces, will detect a face independent whether it is positioned centrally or in a corner of the image [LeCun et al., 2015].

These advantages, availability of huge annotated dataset and the continuously growing amount of computational resources ensured fast advance of CNNs in the field of object detection.

2.1.5 CNN-based Object Detection Algorithms

The basis for successful object detection are informative and distriminative feature representations. Those representations are generated in backbone networks. To improve those feature representations deeper networks with an increased number of parameters have been developed. Important examples are AlexNet, VGGNet, ResNet and GoogLeNet. As winning algorithm at the ImageNet Challange 2012 AlexNet [Krizhevsky et al., 2012] was the beginning of the deep learning era in object detection. It combines different technologies presented in Section 2.1.4: data augmentation, RELU activation functions, dropout and parallel computing with multiple GPUs. VGGNet [Simonyan and Zisserman, 2014] increased drastically the number of layers and parameters compared to AlexNet.

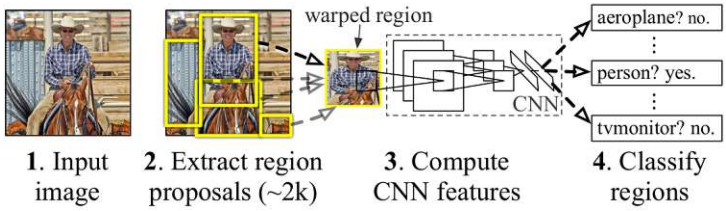


Figure 2.5: RCNN steps to perform object detection. 1. an input image is provided, 2. around 2000 ROIs are extracted, 3. for each ROI features are computed with a CNN, 4. Each ROI is classified based on the features. Figure taken from [Girshick et al., 2014, p. 1]

The most used version is VGG-16 defining 16 uniform layers. With an increasing number of layers and parameters the problem of vanishing and exploding gradients arises. To allow deep networks and therefore be able to learn complex features while keeping meaningful gradient values [He et al., 2016] introduced residual blocks. Another idea to support many layers with many units is to reduce the number of parameters. This can be done with the 1x1 filter kernel [Lin et al., 2013]. They were first implemented in GoogLeNet as Inception modules [Szegedy et al., 2015]. The usage of 1x1 filter kernels allows to increase depth and width of a network while keeping the computational complexity low.

Based on backbone networks different object detection algorithms have been developed. Object detection algorithms can be separated into two general groups, two-stage and one-stage detectors. Two-stage detectors follow the traditional workflow of first scanning the whole scene and in a second step focusing on a set of regions of interest (ROI). One-stage detectors directly map pixel values to bounding box coordinates and classification probabilities in a single step. This fully integrated workflow allows one-stage detectors to achieve real-time inference, while two-stage detectors achieve higher localization accuracy [Zhao et al., 2019b]. In the following first the most important two-stage and one-stage detectors are presented and then their accuracy evaluation and time-analysis is discussed.

Two-stage detectors: As mentioned in Subsection 2.1.3 **RCNN** was the first object detection algorithm leveraging CNNs [Girshick et al., 2014]. It follows the traditional setup as shown in Figure 2.5: first generate region proposals, then run feature extraction based on CNNs and finally perform classification and localization. With this setup RCNN could improve PASCAL VOC 07 mAP_{2D} by 30%. But, due to the multiple steps, employing quite different technologies end-to-end training is no possible. Additionally, RCNN is very expensive in both time and storage and requires fixed size ROI. The last one implying that ROI are clipped or distorted before providing them to the CNN for feature extraction. **SPPNet** (Spatial Pyramid Pooling) [He et al., 2015] solved this by extending RCNN with an additional SPP layer which generates a fixed size representation independent of the size of a ROI. **Fast RCNN** [Girshick, 2015] combines RCNN and SPPNet using a special version of SPP layer called ROI pooling. Additionally it allows to train classification and bounding box regression simultaneously. With this Fast RCNN

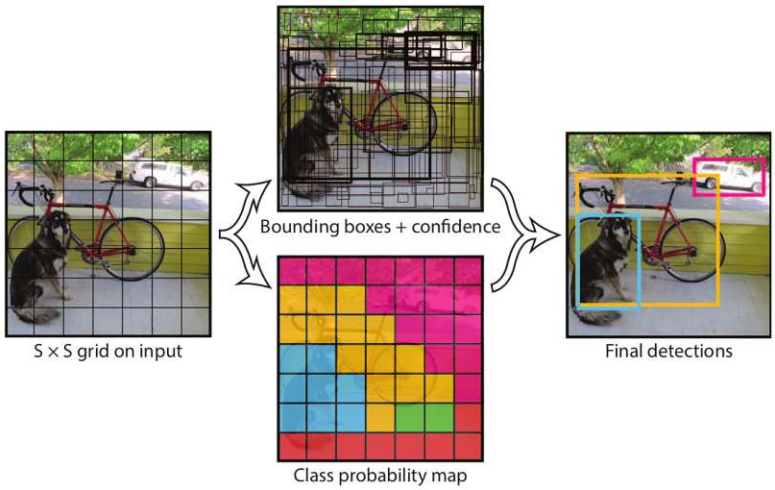


Figure 2.6: General structure of YOLO. For each grid cell, bounding boxes with confidence value and class probabilities are combined to object detections. Figure taken from [Redmon et al., 2016, p. 780]

could further improve PASCAL VOC 07 mAP_{2D} by 20% and it is over 200 faster then RCNN. Still, its bottleneck on detection is the proposal generation. This issue was solved by **Faster RCNN** [Ren et al., 2015] using Region Proposal Network (RPN) for proposal generation based on convolutional layers. With RPN proposal generation is nearly cost free, which improves detection speed by achieving even slightly better accuracy results on PASCAL VOC 07 / 12. Additionally with Faster RCNN the first end-to-end system was developed. A remaining issue with Faster RCNN was its poor detection performance on multi-scale and especially small objects. **FPN** (Feature Pyramid Network) [Lin et al., 2017a] addressed this using not only the top layer for detection but also features from deeper layers. Due to the functionality of CNNs feature pyramids are automatically generated during forward propagation. This idea increased accuracy to state-of-the-art standards and is nowadays also used in backbone networks [Zou et al., 2019]. An example is the FPN RFCN algorithm [Lin et al., 2017a].

One-stage detectors: Due to the handling of different components real-time detection is difficult to achieve with two-stage detector. The solution is to combine the multiple stages into a single regression and classification based system. **YOLO** (You Only Look Once) [Redmon et al., 2016] was the first CNN based one-stage detector. The idea is to split the input image into a SxS grid of regions. Each region is responsible for predicting bounding boxes with their confidence and class probabilities for all objects whose center is located inside the region. Computing this simultaneously for all regions allows to execute the complete object detection task within a single step. With this setup detection times could be decreased extremely, but it also decreases accuracy compared to two-stage detectors. Later version **YOLOv2** [Redmon and Farhadi, 2017] and **YOLOv3**

[Redmon and Farhadi, 2018] especially focus on accuracy issues with dense and small objects. One remaining problem with this version is the decrease of accuracy with increasing IoU thresholds. Recently another update **YOLOv4** [Bochkovskiy et al., 2020], was published leveraging additional features to significantly increase detection accuracy. Starting from YOLO also other one-stage detectors developed. The second group of one-stage detectors is **SSD** (Single Shot MultiBox Detector) [Liu et al., 2016]. Its focus — compared to YOLO first version — was to improve detection accuracy for multi-scale and small objects in dense images. SSD uses anchor boxes with different aspect ratio and size to split the input. Additionally, differently then other networks, SSD detects all object scale on the top layer. Despite important advances one-stage detectors have problems reaching the same accuracy as two-stage detectors. **RetinaNet** [Lin et al., 2017b] addresses this issue by introducing focal loss. The scientists explain the lower accuracy by an imbalance between the extremely high number of candidate location and a relative low number of locations containing objects. This imbalance can be managed with the proposed loss function. RetinaNet reaches two-stage detector accuracy but also detection time increase compared to other one-stage detectors such as YOLO.

2.1.6 Algorithm Comparison

Table 2.2 and Table 2.3 summarize accuracy and speed of different networks evaluated on the MS COCO dev-test dataset. As described above two-stage detectors generally reach a high accuracy while one-stage detectors are faster. RetinaNet and YOLOv4 are an exception to this rule as this one-stage detectors reach the highest accuracy. But, RetinaNet is also the slowest in this comparison.

Comparing the accuracy by object size it can be observed that larger objects are easier to detect than smaller ones. Similarly the accuracy decreases with increasing IoU threshold. AP_{2D-50} reaches the highest results, followed by AP_{2D-75} and finally AP_{2D} which includes AP values for IoU thresholds from 0.5 to 0.95.

It should be mentioned that there a multiple different versions of the presented networks using e.g. bigger or smaller input sizes. This can result in significantly better or worse results in the different categories. An example is YOLOv3-320, YOLOv3-416 and YOLOv3-608, which scales the input image to an input size of 320×320, 416×416, or 608×608 respectively. Depending on the input size AP_{2D-50} ranges from 51.5% to 57.9% and the time ranges from the very low value of 22 ms to 51 ms [Redmon and Farhadi, 2018].

Even though 2D object detection algorithms have reached great accuracy and detection times the resulting bounding boxes can only locate objects in the 2D space. This can be a limiting factor for applications operating in the three dimensional world. Hence, 3D object detection gained more and more attention over the last decade.

Algorithm	Backbone	AP_{2D}	AP_{2D-50}	AP_{2D-75}	AP_{2D-S}	AP_{2D-M}	AP_{2D-L}
Two-stage							
Fast RCNN	VGG-16	19.7	35.9	-	-	-	-
Faster RCNN	VGG-16	21.9	42.7	-	-	-	-
FPN RFCN	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
One-stage							
SSD513	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
YOLOv2-544	Darknet-19	21.6	44.0	19.2	5.0	22.4	35.5
YOLOv3-608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9
YOLOv4-512	CSPDarknet-53	**43.0**	**64.9**	**46.5**	**24.3**	**46.1**	**55.2**
RetinaNet-101-800	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2

Table 2.2: Accuracy comparison between object detection networks on MS COCO test set. Metrics are defined as described in Subsection 2.1.2 besides AP_{2D} which corresponds to AP_{2D}@[.5:.05:.95]. Table created from [Lin et al., 2017b, p. 2987], [Redmon and Farhadi, 2018, p. 3], [Bochkovskiy et al., 2020, p. 11], and [Aziz et al., 2020, p. 170484].

Algorithm	AP_{2D-50}	time [ms]
FPN RFCN	59.1	172
SSD513	50.4	125
DSSD513	53.3	156
YOLOv2-544	44.0	**25**
YOLOv3-608	57.9	51
YOLOv4-512	**64.9**	32
RetinaNet-101-800	57.5	198

Table 2.3: Speed (ms per images) comparison between object detection networks on MS COCO test set. Table created from [Lin et al., 2017b, p. 1], [Redmon and Farhadi, 2018, p. 1], and [Bochkovskiy et al., 2020, p. 11]

2.2 3D Object Detection

2.2.1 Overview

3D object detection is a quite young research domain. The complex and not standardized representation of 3D data, its demanding computational and memory requirements and the need for special sensing devices strongly restricted scientific work. But especially over the last decade one can observe an ever-growing need for 3D solutions. New technologies such as autonomously driving cars, augmented reality, and automated production lines operate in the 3D world. They can not be satisfied with 2D object detection but require accurate 3D information to account not only for the direction of the object but also the distance between sensing device and object. Recent years also saw an increasing availability of computational resources and low-cost measurement devices

as well as important advances in deep learning. Due to these positive developments there is an increasing scientific interest and notable progress in deep learning based 3D object detection [Rahman et al., 2019]. According to the systematic literature review by [Friederich and Zschech, 2020] the number of publications in this domain increased from a single one in 2012, to 36 in 2019.

The basic computer vision problem to solve is the understanding of three dimensional scenes, similar to the 2D case presented in Subsection 2.1.1. 3D object detection can be defined analogously to the 2D domain [Rahman et al., 2019]. Adopting the above cited definition from [Zhao et al., 2019b, p. 3214] it can be stated:

*[... **3D**] object detection aims at locating and classifying existing objects in any one [... **3D scene representation**] and labeling them with [... **oriented rectangular cuboid**] BBs [bounding boxes] to show the confidence and existence.*

Also the computer vision tasks object classification, semantic segmentation and instance segmentation and the differentiation between generic and dedicated object detection can be transferred in an analogous manner [Shen, 2019].

As it can be seen from the adopted definition the resulting bounding box is now three dimensional. Compared to the 2D case additional size, orientation and position parameters are required. In other words 3D compared to 2D bounding boxes have six instead of three degrees of freedom. Therefore a minimum of seven parameters are required to describe 3D bounding boxes [Friederich and Zschech, 2020]. A widely used method is to store the center and its offset [Song and Xiao, 2016]. It defines a bounding box as its center coordinates (x, y, z), their offsets (Δx, Δy, Δz) and the yaw angle ($\Delta\theta$). The bounding box is considered parallel to the $x - y$ plane, therefore pitch and roll angle are set to zero. Other encoding methods include storing absolute coordinates of all 8 corners [Chen et al., 2017] or defining four coordinates and two heights [Naseer et al., 2018].

The other highlighted difference between 2D and 3D object detection — presented in the definition — is the input data. Instead of images any type of 3D scene representations can be used. This representation is often defined by the sensing technology. Commonly used technologies are stereo cameras, LIDARs, RGB-D cameras, Radars and Ultrasonics. Stereo cameras are standard RGB camera capturing the color, texture and appearance of objects from two orientations simultaneously. Their disadvantages is that they require external light — as they are a passive system —, are weather dependent and the derived depth accuracy decreases exponentially with distance. LIDARs (Light detection and ranging) are active sensors which emit infrared light. The distance to objects is calculated by measuring the elapsed time between emitting and receiving a signal. As active sensors LIDARs are illumination independent and are hardly affected by different weather conditions. Due to their signal strength they can measure accurate absolute distances up to 200 meters. The result is a sparse point cloud without color or texture information. RGB-D cameras are a combination of RGB cameras and ToF (Time of Flight) depth cameras. Therefore they return color, texture and depth information per pixel. As they provide only accurate depth information up to 8 meters they are mainly used in

indoor applications. Radars (Radio Detector And Ranging) actively emit radio waves and are therefore also illumination and weather independent, but they can interfere with other systems and only provide a low resolution due to their long wavelength. Similar considerations apply for ultrasonic sensors which emit high frequency sound waves. Ultrasonic is often used for close-range object detection [Rahman et al., 2019], [Arnold et al., 2019]. After data acquisition 3D data can be represented in a number of different formats. Popular formats are stereo images, depth images, voxel-grids, unstructured point clouds and polygonal meshes [Singh et al., 2019].

Each acquisition technology and data format has different properties and may therefore be better or worse for specific applications or even whole domains. One distinction is especially popular, the one between indoor and outdoor scenes. An example are bird-eye-view images (BEVs). In an outdoor scene, for instance a traffic situation, they are a useful data representation as it is easy to identify instances, their location and extend. This is possible because objects are hardly above one another. This assumption is not valid for indoor scenes like an office room where a book shelf can be above an office desk with monitor. Due to the large amount of possible 3D representations algorithms developed differently and domain driven. This quite independent, domain driven development resulted in heterogeneous 3D object detection approaches [Rahman et al., 2019].

2.2.2 Datasets and Metrics

To compare 3D object detection algorithms similar to the 2D domain a set of standard dataset and metrics evolved.

The difference to the 2D domain is that acquisition and annotation of 3D data can be an expensive and demanding task as special tools and software solutions are required. But, deep learning based algorithms strongly depend on the availability of large annotated datasets. Publicly available annotated datasets are therefore not only important to compare detection algorithms but also to support and advance the development of new ones [Rahman et al., 2019].

Datasets

As mentioned above 3D object detection evolved in a very domain specific manner. This can also be seen in the presented standard datasets as they can generally be grouped into indoor and outdoor datasets. Due to the especially high impact of autonomous driving outdoor datasets often focus mainly on this application [Geiger et al., 2012]. Over the last years also other dedicated object detection datasets have been created, focusing e.g. on industrial scenes (see e.g. [Drost et al., 2017], [Hodan et al., 2017]) or object parts (see e.g. [Yi et al., 2016]). Due to application and sensor driven development one can observe that indoor datasets mainly rely on RGB-D and sometimes 3D models and outdoor datasets often also include LIDAR point clouds (see Table 2.4). In the following five popular datasets are presented in more detail.

Dataset	Scene type	Sources	# Classes	train	validation	test
NYU-Depth V2	Indoor	RGB-D	40	795	-	654
SUN RGB-D	Indoor	RGB-D	800	2,666	2,619	5,050
KITTI	Urban(Driving)	RGB & LIDAR	8	7,481	-	7,518
FAT	Indoor	RGB & 3D models	21	61,500	-	-
nuScenes	Urban(Driving)	Camera & LIDAR	23	1.4M	-	-

Table 2.4: Five popular 3D object detection datasets with their statistics. Table adopted from [Rahman et al., 2019, p. 2950].

NYU-Depth V2 [Silberman et al., 2012][5]: One of the first 3D datasets published in 2012 was NYU-Depth V2. It contains 1449 densely labeled pairs of aligned RGB and depth images which were captured by Microsoft Kinect. Those pairs show 464 different scenes from three cities. They are classified into 40 indoor categories and split in 795 training and 654 test samples.

SUN RGB-D [Song et al., 2015][6]: The first large-scale 3D datasets — comparable to PASCAL VOC for 2D data [Everingham et al., 2012] — was SUN RGB-D. It is composed of three other datasets: NYU-Depth V2 [Silberman et al., 2012], Berkeley B3DO [Janoch et al., 2013] and SUN3D [Xiao et al., 2013] and was captured by four different sensors. In total it contains 10,335 RGB-D images with both 2D and 3D annotations, including 146,617 2D bounding boxes and 64,595 3D bounding boxes. There are 47 scene and 800 object categories. It should be mentioned that the frequency of single object categories might be quite different. There are nearly 20,000 annotation for the most frequent object category chair, but most categories like plant, tv or bench have less then 500 samples. To reduce the number of categories, often either the 19 categories with the highest number of annotations, or the so called 10-class set (bathtub, bed, bookshelf, chair, desk, dresser, nightstand, sofa, table, toilet) are used for algorithm evaluation. Due to the diversities of different sensors and object category frequency, the data is carefully divided into training, validation and test set. SUN RGB-D enabled important progress in different computer vision tasks [Rahman et al., 2019].

FAT (Falling Things) [Tremblay et al., 2018][7]: A completely synthetic dataset for object detection and 3D pose estimation is FAT. It was generated by placing 3D household object models in front of virtual backgrounds. It provides aligned mono, stereo and depth images annotated with 3D pose, per-pixel category segmentation, and 2D as well as 3D bounding boxes. Over 60k images annotated with 21 categories are available. The dataset is composed of two parts, the first half of the images feature exactly one object, the second half show between 2 and 10 objects sampled randomly. With this another even larger but synthetic indoor dataset was created.

[5] https://cs.nyu.edu/ silberman/datasets/nyu_depth_v2.html, accessed 9-April-2021
[6] http://rgbd.cs.princeton.edu, accessed 9-April-2021
[7] https://research.nvidia.com/publication/2018-06_Falling-Things, accessed 9-April-2021

KITTI [Geiger et al., 2012][8]: KITTI is one of the most important and popular datasets used for autonomous driving. It is composed of stereo color images, LIDAR point clouds and GPS coordinates, all synchronized in time. As a number of different input data types are provided KITTI can be used for a wide range of computer vision tasks. For object detection in total 7,481 training and 7,518 test samples are provided annotated with a total of 80,256 objects. It further divides the data into easy, moderate and hard samples based on object size, occlusion and truncation level. As it focuses on the urban driving scenario its main categories are car, pedestrian, and cyclist though there is a strong category frequency imbalance, as there are 75% cars, 15% pedestrians and only 4% cyclists annotations. Moreover, the data was acquired mainly during daytime and sunny conditions. To overcome these restrictions a synthetic extension called Virtual KITTI [Gaidon et al., 2016] was proposed.

nuScenes [Caesar et al., 2020][9]: One of the largest currently available datasets in the autonomous driving domain is nuScenes. Compared to the popular KITTI datasets it contains significantly more samples. Data was captured with six cameras, five RADAR and one LIDAR all having a 360 degree view. This results in approximately 1.4M images, 390K LIDAR point clouds, 1.4M RADAR scans and 1.4M bounding boxes annotating 23 categories. Data was acquired in two cities during day and nighttime and under various weather conditions.

Compared to datasets presented in Subsection 2.1.2 3D datasets are currently still a lot smaller, contain less scenes and also annotated objects, but cover a wide range of input data formats.

Metrics

Comparison between algorithms is based on a set of evaluation criteria, which are defined similarly in the 3D and 2D domain. A general distinction is made between detection speed and accuracy measurements. The first is defined in a complete analogous manner as in Subsection 2.1.2. The latter requires some extension compared to the 2D case to cover the additional dimension.

The standard accuracy evaluation metric is also in the 3D domain Average Precision (AP_{3D}) per category (Equation 2.3) and mean Average Precision (mAP_{3D}) (Equation 2.4). Both are based on 3D volume Intersection over Union (IoU) [Song and Xiao, 2014]. As mentioned above 3D bounding boxes are assumed to be oriented parallel to the x-y plane but may be rotated around the z-axis, which is described as the *orientation* of a bounding box. Due to the additional degrees of freedom it is more difficult to get perfectly matching predicted and true bounding boxes in the 3D domain. Therefore usually the IoU threshold to accept a bounding box is lower — in the 3D domain a standard value of 0.25 is used, whereas in the 2D domain a standard value of 0.5 is used [Rahman et al., 2019].

[8] http://www.cvlibs.net/datasets/kitti, accessed 9-April-2021
[9] https://www.nuscenes.org, accessed 9-April-2021

In more detail there are multiple extensions and modification for AP_{3D} and mAP_{3D} to better describe the 3D domain. Due to the used data representation or projections also 2D average precision are used. An example for a 2D based metric is Average Orientation Similarity (AOS). For this the orientation similarity per recall is calculated as AP_{2D} in the image plane weighted with the cosine similarity between true and predicted orientation. AOS is then the average over the orientation similarities [Geiger et al., 2012]. Especially for outdoor scenes a better description can be achieved by using AP_{BV} bird's eye view metric [Chen et al., 2017]. Employing AP_{3D} removes distortion effects because of projection and properly accounts for 3D bounding box size and localization. The disadvantages of pure AP_{3D} is its insensitivity to the orientation as long as the overlapping area between predicted and true bounding box are large enough. This issue is addressed with the Average Heading Angle (AHS) metric. It is an extension to AOS using AP_{3D} instead of AP_{2D}, calculating the orientation similarity as AP_{3D} weighted cosine similarity between true and predicted orientation [Ku et al., 2018].

2.2.3 3D Object detection by input data

Based on publicly available datasets 3D object detection systems evolved. As stated in Subsection 2.2.1 the system gets a 3D scene representation as input and outputs 3D bounding boxes, class labels and their confidence. Based on the different types of 3D scene representations 3D object detection algorithms can be grouped into image-, volumetric- and fusion-based methods [Rahman et al., 2019].

Image-based methods work on monocular images without depth information. To predict a 3D bounding box normally first a 2D bounding box is predicted, then the third dimension is estimated based on neural networks [Chen et al., 2016], geometric constraints [Mousavian et al., 2017] or 3D model matching [Xiang et al., 2015]. The opposite to this quite data limited set of methods are fusion-based methods which use a combination of 2D and 3D data. Fusion-based methods benefit from using both color and texture information provided by images as well as accurate depth measurements available in point clouds. Examples are MV3D [Chen et al., 2017], AVOD [Ku et al., 2018] and F-PointNet [Qi et al., 2018]. Both image- and fusion-based methods are not covered in detail here as this thesis focuses on the comparison between pure 2D and 3D based algorithms. In following the different volumetric methods are presented.

Volumetric-based methods solely use 3D information as input. Based on the exact input data representation they can be further divided into three subcategories. First, the 3D data can be projected to a 2D plane. Those methods are summarized as projection or view-based methods. Second, data can be stored in a structured 3D format — a voxel-grid. Third, raw unstructured point clouds can be used [Singh et al., 2019].

View-Based Methods

Using view-based methods 3D data is first projected to a 2D plane. Popular options are front view, top view (also called bird's eye view, BEV) or range view. The idea of

converting 3D to 2D data is motivated by the good object detection results achieved with state-of-the-art 2D algorithms and the availability of large-scale datasets and benchmarks as presented in Section 2.1. The projected data can then simply be handled like any 2D input. In other words it can be processed using conventional deep neural networks. To obtain a 3D bounding box from 2D results, they are regressed in position and dimension [Arnold et al., 2019].

VeloFCN [Li et al., 2016] and **LMNet** [Minemura et al., 2018] are two examples of frontal view cylindrical projections. Both use a fully convolutional network for detection but VeloFCN inputs 2D depth maps and LMNet five different frontal-view representations: reflection, range, forward, side, and height. To avoid occlusion problems immanent to frontal view projections, recently bird's eye view representations received a lot of attention. Using BEVs object length and width as well as the position on the ground plane can easily be obtained. Two examples are **DoBEM** [Yu et al., 2017] and **BirdNet** [Beltrán et al., 2018]. Again the 3D representation is encoded into a three channel 2D representation. DoBEM transforms it into an elevation map, namely maximum, median and minimum height, BirdNet uses height, intensity and density.

All methods presented so far are two-stage detectors, so first a set of region proposals is derived, then a refinement step calculates the final classification scores and bounding boxes. To improve inference time a set of one-stage detectors evolved. These detectors directly map the input to the final classification scores and bounding boxes. Up to now only BEV based one-stage detectors are available. **PIXOR** [Yang et al., 2018] uses height-encoded BEVs similar to DoBEM and outputs pixel-wise predictions in a single step, but is applies the assumptions that all objects are one the ground. **Complex-YOLO** [Simony et al., 2018] and **YOLO3D** [Ali et al., 2018] are both based on the popular 2D YOLOv2 [Redmon and Farhadi, 2017] algorithm focusing especially on efficiency. Both employ BEVs, but Complex-YOLO extends the architecture by a specific complex regression strategy and YOLO3D extends YOLOv2's loss function to cover the additional parameters required for 3D object detection.

Generally, 3D information stored in a 2D format can reduce memory requirements and computational costs. Additionally it enables the usage of existing 2D algorithms on 3D data or at least simplifies implementation of new algorithms. But, view-based representations often suffer from poor orientation angle regression. Currently this group especially focuses on BEV based algorithms which can be very useful for outdoor applications such as autonomous driving but are not applicable for indoor environments featuring multiple objects above each other [Rahman et al., 2019]

Voxel-Grid-Based Methods

To overcome the limitations of 2D representations 3D voxel-grid representations can be used. In that case data is stored in a regular 3D grid, called voxel-grid. Each voxel is either defined by binary occupancy or continuous point density and can have a number

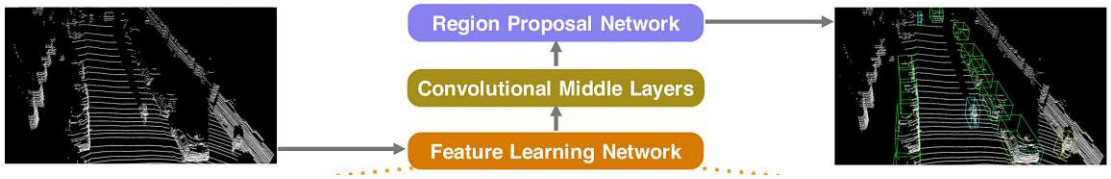


Figure 2.7: VoxelNet architecture: Derivation of 3D bounding boxes from raw point cloud leveraging a voxel-grid. First, the feature learning network partitions the input point cloud into a voxel-grid and generates a representative vector per voxel describing the shape. Second, the convolutional middel layer aggregates spatial context. Third, the region proposal network calculates the final 3D detections. Taken from [Zhou and Tuzel, 2018, p. 4491s].

of attributes or dimensions. This regular structure allows to apply concepts developed for 2D data such as CNNs by simply adding one dimension [Rahman et al., 2019].

An example is **3DFCN** [Li, 2017] which is based on VeloFCN [Li et al., 2016]. It reuses 2D mechanisms on a 3D grid. 3DFCN is a single-shot detector which inputs a 4D binary encoded voxel-grid with length, width, height and channel dimension and directly outputs objectness and bounding boxes. Although 3DFCN is a single-shot detector the time performance is limited due to the use of expensive 3D convolutions.

A more efficient algorithm is **Vote3Deep** [Engelcke et al., 2017] which is a modification of **Vote3D** [Wang and Posner, 2015]. Vote3D discretizes the 3D representation into fixed resolution voxels, calculates a fixed size handcrafted feature vector per voxel and then runs object search which is realized as sliding window with a Support Vector Machine (SVM). Due to the feature-centric approach Vote3D is already quite efficient. To improve it even further Vote3Deep replaces the SVM with a 3D CNN. Vote3Deep exploits the sparsity to point clouds by using a sparse CNN and L1 regularization together with RELU to preserve the sparsity. Its downside is that object sizes are fixed which limits detection performance.

A neural network example designed for indoor environments is cloud of oriented gradients (**COG**) [Ren and Sudderth, 2016] proposing the Manhatten voxel representation. It was later improved to latent support surfaces (**LSS**) [Ren and Sudderth, 2018]. The main downside of these models is their long inference time of 10 to 30 minutes.

A completely end-to-end trainable algorithm is **VoxelNet** [Zhou and Tuzel, 2018] depicted in Figure 2.7. It uses raw point clouds as input and does not require any manual feature engineering. It can be separated in three steps: 1. feature learning network, 2. convolutional middle layers, 3. region proposal network. In the first step the point cloud is converted to a voxel-grid and a feature vector - called voxel feature encoding - is calculated per voxel. This is based on PointNet [Qi et al., 2017a] described in more detail in the next part. In the second step features are aggregated and additional context information is added. In the last step the final 3D object detection results are obtained. Similar to Vote3Deep VoxelNet utilize the sparsity of the data but also suffer from

the high computational costs of 3D convolutions. **SECOND** [Yan et al., 2018] further improves VoxelNet. It tries to reduce computational costs by converting the 3D data to a 2D representation in the convolutional middel layers. Additionally, a new angle loss regression strategy is used to improve the orientation estimation.

As mentioned above voxel-grid-based methods have the advantages of explicitly encoding and preserving 3D shape information. Moreover, the structured representation allows to apply the same principals used in 2D object detection. But, the structured data representation is also a disadvantages. Most voxels are empty but still require storage space and need to be processed which reduces efficiency. Generally, voxel-grid-based methods employ 3D convolutions which drastically increase computational costs, this in turn constraints the processable resolution [Arnold et al., 2019].

Point Cloud-Based Methods

Typical CNNs require a regular structure, input data must have a fixed size. To use such CNNs on point clouds often a transformation is applied to the data to get either a 2D projection (view-based) or a 3D regular grid (voxel-grid-based). Any of those transformations leads to a loss of information. To overcome this drawback it is possible to work directly on raw point clouds [Arnold et al., 2019].

The seminal idea was presented in **PointNet** [Qi et al., 2017a] which focuses on object classification and part segmentation. PointNet uses raw point clouds as input. For all n input points local features are inferred. This is achieved by point-wise transformation with fully connected layers. Then this local feature descriptors are aggregated to a global one with a max-pooling layer which works as symmetric function and ensures independence of the point order. The main disadvantage of PointNet is that it misses local structures as it strongly focuses on global features. **PointNet++** [Qi et al., 2017b] addresses this issue by explicitly learning local structures. The n input points are split into overlapping sets. For each set local features are extracted. In a hierarchical approach simple features are then grouped into more complex ones.

PointNet and PointNet++ are object classification but not object detection algorithms. Over the last years a number of methods evolved utilizing PointNet. One example is VoxelNet presented before which uses PointNet per voxel. Most published methods require both image and point cloud data. Examples are F-PointNet [Qi et al., 2018], PointFusion [Xu et al., 2018] and RoarNet [Shin et al., 2019]. Due to the requirement of multiple input data types they are categorized as fusion-based methods and are not relevant for this thesis.

To the authors best knowledge the only purely 3D data based network leveraging PointNet is **VoteNet** [Qi et al., 2019]. Next to PointNet VoteNet adopts Hough Voting. VoteNet's basic architecture can be split into four main steps presented in Figure 2.8. First, find points of interest. Second, generate votes — basically new points close to the object center augmented with an additional feature vector. Third, aggregate vote clusters based on features. Fourth, return object proposals as classification scores and 3D bounding

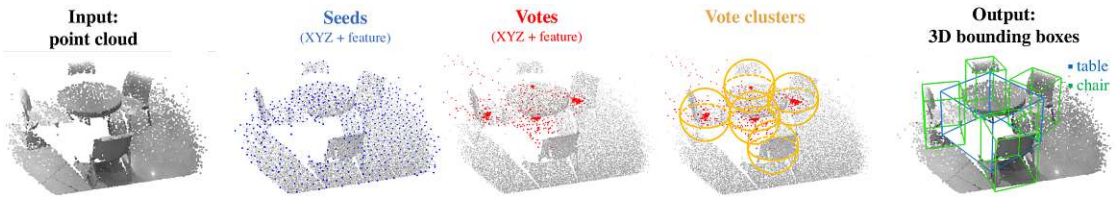


Figure 2.8: VoteNet architecture: First, the n input points are sampled to m seed points leveraging PointNet++. Then the voting model generates one vote per seed point. The votes are grouped to clusters which are in the following used to create object proposals. Figure taken from [Qi et al., 2019, p. 3].

boxes. All of these steps are combined into an end-to-end trainable network. Recently an extension was proposed called ImVoteNet [Qi et al., 2020] combining point cloud data with images.

Point cloud-based methods fully leverage the sparsity of point clouds. Calculations are solely performed on measurements (points) and not in empty space such as for voxel-grid-based methods. Moreover, 3D geometric details are preserved and used. New concepts such as PointNet have the potential to overcome traditional requirements for structured data and provide mechanisms which are better applicable to this irregular input data [Qi et al., 2019].

2.2.4 Algorithm Comparison

View-, voxel-grid- and point cloud-based methods can all be evaluated on the datasets presented in Subsection 2.2.2. Due to the domain driven development and the different requirements defined by applications it is not meaningful to evaluate every method on every dataset. Most algorithms are evaluated either on KITTI (Table 2.5) or SUN RGB-D (Table 2.6).

View-based methods achieve quite poor results on 3D object detection KITTI test set. This can be explained as many bird's eye view based methods primary focus on KITTI BEV test set results during development not presented here. LMNet is with 6ms the fastest presented method but also the one with the lowest mAP_{3D}. Most results are available for voxel-grid-based methods. Vote3Deep achieves the highest accuracy with a mAP_{3D} value of 66.15 on KITTI test set but it is also the slowest on this dataset. SECOND reaches good results both in respect to accuracy and speed being the second best in both categories. The only point cloud-based algorithm VoteNet reaches the best results in both accuracy and speed on SUN RGB-D.

A direct comparison between 2D (Table 2.3 and Table 2.3) and 3D algorithms (Table 2.5 and Table 2.6) is not possible as different datasets and metrics are used. But one can observe that 2D and 3D accuracy values already have the same magnitude while 2D algorithms are still faster. This similarities and differences are especially interesting for applications providing access to both data sources such as Augmented Reality.

Algorithm	car	pedestrian	cyclist	mAP_{3D}	Speed [s]
view					
VeloFCN	50,20	-	-	-	1.000
LMNet	15.24	11.46	3.23	9.98	**0.006**
DoBEM	6.95	-	-	-	0.600
BirdNet	13.41	12.22	14.22	12.56	0.110
voxel-grid					
Vote3D	49,12	37,98	33,76	40,28	0.500
Vote3Deep	69,42	**58,78**	**70,26**	**66,15**	1.100
VoxelNet	65.11	33.69	48.36	49.05	0.230
SECOND	**74.33**	43.64	57.09	56.69	0.038

Table 2.5: Performance comparison of state-of-the-art 3D object detection based on 3D KITTI test set. Table created from [Rahman et al., 2019, p. 2958], [Engelcke et al., 2017, p. 1360], and [Minemura et al., 2018, p. 32]

Algorithm	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	mAP_{3D}	Speed
voxel-grid												
COG	58.3	63.7	31.8	62.2	**45.2**	15.5	27.4	51.0	51.3	70.1	47.6	10-30m
LSS	**76.2**	73.2	**32.9**	60.5	34.5	13.5	30.4	60.4	**55.4**	73.7	51.0	10-30m
point cloud												
VoteNet	74.4	**83.0**	28.8	**75.3**	22.0	**29.8**	**62.2**	**64.0**	47.3	**90.1**	**57.7**	**0.10s**

Table 2.6: Performance comparison of state-of-the-art 3D object detection based on 10-class evaluation on SUN RGB-D dataset. Table created from [Rahman et al., 2019, p. 2958] and [Qi et al., 2019, p. 6]

2.3 Object Detection with Augmented Reality

2.3.1 What is Augmented Reality?

Ever since the first computers were available the goal has been to improve the human computer interaction to make it more intuitive. Digital interaction should be as simple as interacting with real objects. Augmented Reality (AR) realizes this idea by removing the separation between digital and real world. This allows enhanced communication, new information presentation possibilities and intuitive digital interaction [Billinghurst et al., 2015].

Augmented Reality combines the real and virtual world. In detail it can be defined by three key requirements: first, it is a combination of real and virtual content, second, it works interactively in real-time, and third, it is registered in 3D [Azuma, 1997]. The idea is to superimpose virtual objects — so-called *holograms* — onto real world-objects. This should enhance the real world with additional information required or requested by the user [Van Krevelen and Poelman, 2010]. The aforementioned requirements also

define the main three technical components, namely, a display to visualize virtual objects onto the real world, user interaction in real-time, and a tracking system to derive user position and orientation in the real world [Billinghurst et al., 2015].

AR can be categorized by the used display. The most common ones are head mounted displays (HMD), wearable devices such as glasses or helmets. This group can be further classified into optical and video see-through systems. Optical see-through systems present the user the real world through a half-transparent mirror which allows to additionally reflect virtual objects into the user's eyes. Video see-through systems first digitize the real world, then combine it with virtual content and finally show the user this combination on an opaque display [Azuma, 1997]. Another more recent display option are mobile screens, such as smartphones or tablets. Similar to video see-through HMDs the real world is captured, combined with virtual objects and then visualized on the screen [Nishihara and Okamoto, 2015]. The third option are Spatial Augmented Reality systems (SAR) which use projectors to display virtual content directly onto real objects [Azuma et al., 2001].

To properly superimpose virtual onto real objects the precise user position and orientation in the real world are required. The simplest method to achieve this is by using markers in the real world. The markers are identified by the camera and their pattern is compared to previous ones to estimate position and orientation [Khan et al., 2015]. Physical markers may not always be feasible. Two methods independent of physical markers are Natural Feature Tracking (NFT) and Simultaneous Localization and Tracking (SLAM). NFT detects characteristic points instead of markers in images to estimate the pose in real-time [Fraga-Lamas et al., 2018]. SLAM creates a 3D feature map which is used to estimate the pose based on the currently visible part. This information then also enhances the initial map [Billinghurst et al., 2015].

Since the first reference of Augmented Reality by [Thomas and David, 1992] AR systems have proven their helpfulness in main domains. Examples range from education [Dünser, 2008], [Bacca Acosta et al., 2014] over medicine [Park et al., 2020a], [Birkfellner et al., 2002] to manufacturing [Funk et al., 2017], [Chu et al., 2020]. Different studies showed that AR systems can support learning new tasks, reduce execution times, and improve product quality [de Souza Cardoso et al., 2020]. An important base task — required in all aforementioned domains — is object detection.

2.3.2 Object Detection Systems

Most AR based systems must capture and understand a set of objects in their surrounding environment. Similar to user tracking, such technologies often rely on physical markers positioned in the real world. This is a limiting factor especially in dynamic environment such as industrial manufactures [Park et al., 2020c] or in uncontrolled outdoor environment where it is not possible to properly place markers [Rao et al., 2017]. A number of AR systems already leverage other technologies such as object detection. [Wang et al., 2013] tracked index finger and thumb without additional markers to support manual assembly design and [Radkowski, 2016] could tracked a single object based on point cloud matching.

In recent years a number of studies successfully directly combined AR and object detection and evaluated the results. They can generally be grouped into completely embedded systems and setups separated into a client and server. Many AR systems such as wearable or mobile devices only provide limited computational resources which limits the use of expensive deep neural networks.

Embedded systems [Rao et al., 2017] therefore leverage efficient and small backbone networks such as MobileNet [Howard et al., 2017] and fast object detection algorithms such as SSD [Liu et al., 2016] or YOLOv2 [Redmon and Farhadi, 2017]. Most introduced systems are designed for mobile devices and integrate additional components to improve their results. [Lee et al., 2019] presented DeepMobileAR, an Android application. It combines SSD-MobileNet object detection with visual SLAM. SLAM handles virtual object pose calculation, camera matrix estimation and mapping of the surrounding environment. [Mahurkar, 2018] integrated TinyYOLOv2 (lightweight YOLOv2 implementation) and AR into an iOS application. [Rao et al., 2017] developed an Android application which utilizes a lightweight SSD implementation and enhances the detections with additional data from Global Positioning System, Inertial Measurement Unit and magnetometer to properly localize objects in the real world.

Client-server systems extend the limited on device resources of AR by connecting it to a server or cloud. This allows the usage of more complex approaches and avoids a possible lack of detection accuracy. The drawback of this setup are communications delays between client and server. A combination of the HMD Microsoft HoloLens [Microsoft, 2021a] and YOLOv2 running on an external server was presented by [Eckert et al., 2018]. It is a system for the blind to localize wanted objects. [Farasin et al., 2020] achieved even real-time responses by integrating high accuracy object detection (Faster-RCNN [Ren et al., 2015], R-FCN [Dai et al., 2016] and SSD [Liu et al., 2016]) in the cloud with fast object tracking on the device (Microsoft HoloLens). A combination of AR, object detection and pose estimation presented by [Kästner et al., 2021] showed positive effects such as reduced error rates and increased time efficiency.

Images captured with wearable AR devices often have a very different perspective then images available in standard datasets such as MS COCO [Lin et al., 2014]. In standard datasets the image orientation is often object dependent, elongated objects are often captured in landscape. To compensate for this [Li et al., 2020] presented a system which accounts for the orientation and also uses the additional available scale information provided by AR systems. These steps improved generic object detection accuracy by 12%. This shows two things. First, available datasets are not optimized for AR applications, and second, already simple 3D information has the potential to significantly improve performance. But to the author's best knowledge AR has only been combined with 2D and not 3D object detection even though a number of systems provide 3D information. An interesting example in this category is Microsoft HoloLens.

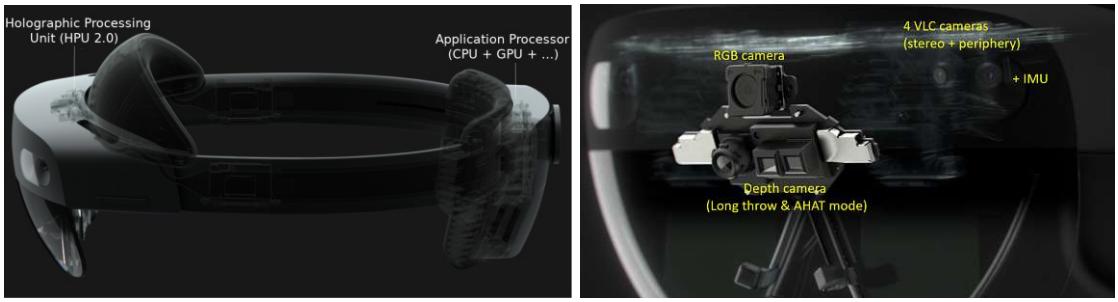


Figure 2.9: Microsoft HoloLens 2. a) Overview with processors. b) Available sensors. Adopted from [Ungureanu et al., 2020, p. 1 / p. 3].

2.3.3 Microsoft HoloLens

Microsoft published in 2016 their first AR system called Microsoft HoloLens [Microsoft, 2021a]. In 2019 its successor HoloLens 2 [Microsoft, 2021b] was announced. Both versions have been adopted in different domains, such as surgery (see e.g. [Schneider et al., 2021], [Park et al., 2020b]), engineering (see e.g. [Zhang et al., 2019]) and education/training (see e.g. [Asgary et al., 2020], [Serrano Vergel et al., 2020]).

Specification

HoloLens 2 is an enhancement of HoloLens first generation with new and improved features, such as a larger field of view and fully articulated hand and eye gaze tracking. Both are head mounted wearables which allow to move freely [Microsoft, 2021b]. As this thesis focuses on HoloLens 2 in the following it is described in more detail.

Two processors are available (see Figure 2.9a). In the front part second generation Holographic Processing Unit (HPU 2.0) is located. It is designed to handle all low-power, real-time computer vision tasks on the system, such as head, hand and eye gaze tracking as well as spatial mapping. In the rear part a Qualcomm SnapDragon 850 is mounted. As all computer vision tasks are handled by the HPU this is a pure application processor [Terry, 2019]. HoloLens 2 comes with 4 GB memory and 64 GB storage. Connectivity is possible over Wi-Fi, Bluetooth or USB [Microsoft, 2021b]. The system is equipped with a depth, RGB and four gray scale cameras, called visible-light tracking cameras (VLC). Additionally, an Inertial Measurement Unit (IMU) and a five channel microphone are available (see Figure 2.9b) [Ungureanu et al., 2020].

Available Data Sources

As mentioned in Section 2.1 and Section 2.2 object detection can be performed based on different input data types. HoloLens offers 2D and 3D data describing its surrounding environment [Ungureanu et al., 2020], [Khoshelham et al., 2019].

2D object detection requires either RGB or grayscale images as input. 3D object detection is either also based on images, point clouds or depth information. RGB data is accessible either as video stream or single picture covering a field of view of 40° × 25°. Different resolutions are available from 896 × 504 to 1408 × 792 pixels all with a 30 fps [Hübner et al., 2020].

To extend available data sources HoloLens 2 provides a dedicated Research Mode API which allows access to the different sensors. The four grayscale VLC are available at 30 fps. The depth camera operating in two modes, either at a high framerate of 45 fps in near-depth (used for hand tracking) or at a low framerate of 1 to 5 fps in far-depth (used for spatial mapping). Additionally, the same two depth modes are also available for the infrared system which is independent of lightning condition. The drawback of using these streams is that the *development mode* needs to be activated on the device. Therefore application which utilize the Research Mode API can not be used in production systems but only in a research environment [Ungureanu et al., 2020].

Another possibility to access 3D data is the generated spatial mesh. HoloLens automatically maps its surrounding environment with the so-called *spatial mapping capability*. This capability calculates a 3D sparse triangular mesh of the environment. This is required to allow a proper combination of the real and virtual world and to locate itself within it. The spatial mesh is used implicitly when for example interacting with a hologram. To fine tune the spatial mesh to the user's needs it can be calculated at different resolutions. It must be considered that a higher resolution mesh requires a more expensive calculation which may constraint response times [Zeller, 2018].

The spatial mapping algorithm is largely unpublished and part of Microsoft's proprietary software. Likely it is a combinations of the SLAM algorithms developed for KinectFusion [Izadi et al., 2011] and enhancements such as large-scale reconstruction based on voxel-hashing [Nießner et al., 2013] and RGB-D camera relocalization [Glocker et al., 2014] which improve scalability and robustness. The mesh is calculated based on depth and VLC tracking data, RGB images are not used [Hübner et al., 2020]. Therefore HoloLens has problems mapping objects with a reflecting surface, as a single instance may appear differently depending on the viewing angle. In particular, dark, translucent and shiny objects are problematic [Zeller, 2018].

Due to the simple accessibility of the spatial mesh HoloLens can also be considered a simple mapping tool. Compared to laser scanners or range cameras HoloLens is more flexible and efficient as it can easily be moved around and view objects from different directions and angles. Therefore it is especially suitable for complex and changing environments. Additionally, already HoloLens first generation achieves a good accuracy of about 5 cm compared to a terrestrial laser scanner in indoor environments [Khoshelham et al., 2019], [Hübner et al., 2020]. The spatial mapping capability has already been exploited for building modeling [Hübner et al., 2019] and roboter setup configuration [Puljiz et al., 2020]. But the available 3D data has not yet been leveraged for object detection.

2.4 Contribution of this work

The previous sections describe the current status of 2D and 3D object detection as well as the concept of AR. For both object detection domains theoretical background is given, datasets and metrics are presented, and state-of-the-art algorithms are described and compared by accuracy and inference time. Based on these previous findings, this thesis focuses on the combination of 2D and 3D object detection with AR. In detail the following three open topics are addressed:

1. **Creation of a comparable AR 2D / 3D datasets**: There are already a number of 3D datasets available. Those often also contain 2D image data, but compared to 2D datasets they are still limited in size. SUN RGB-D combined multiple other datasets to achieve a size comparable to the popular PASCAL VOC. But PASCAL VOC appears small compared to recent 2D datasets such as MS COCO. One of the largest and most recent 3D dataset, FAT, is synthetically created and does not contain real pictures. Additionally, due to the domain driven development of 3D object detection available data either covers solely indoor scenes or in the outdoor domain has a strong focus on the autonomous driving application. But there is no dataset containing both indoor and outdoor scenes. Hence, there is no real generic dataset available. Moreover, there is none for the AR domain, depicting the unusual perspectives especially popular with wearable AR devices. To fill this gap an AR generate dataset covering commodity items in both indoor and outdoor environment is created and introduced. In the following it is called *AR-2/3*.

2. **3D object detection with AR**: Already a number of systems exist combining 2D object detection with AR. Although recent research showed the usefulness of 3D information for AR object detection 3D AR data has not yet been used for object detection. Some AR devices such as Microsoft HoloLens even provide 3D data automatically. Especially for technologies operating in the 3D world 2D bounding boxes are not always sufficient but the accurate 3D extend of objects is required. AR scene understanding, navigation, and virtual interactions may depend on 3D information. In this thesis AR is combined with a representative 3D object detection algorithm.

3. **Comparison between 2D and 3D object detection**: 2D object detection is a well-research field nowadays leveraging the power of deep neural networks and the availability of large, public datasets. In comparison is the 3D pendant a quite young research area originating from the need of 3D localization in specific domains. But recent years saw significant progress in 3D object detection. Different data representations and algorithms have been used to constantly improve accuracy and speed. The main advantage of 3D over 2D object detection is that it provides information about an additional dimension. An open question is how 3D algorithms perform compared to 2D ones. Up to now only rough estimates have been possible as no comparable 2D / 3D dataset has been available. Based on the newly created

dataset mentioned above it is possible to compare the maybe more elaborated 2D techniques with the more recent and more informative 3D algorithms. For this next to the setup of a 3D object detection system also a 2D algorithms is implemented and results are compared. In detail accuracy and inference time are evaluated. This highlights shortcomings and advantages of current algorithms in the context of a different number of applied dimensions.

All data acquired and processed throughout this thesis is available under `//GEO/geoinfo/Data/Sophie/masterarbeit_backup` and licensed under the terms of Creative Commons Attribution 4.0 International (CC BY 4.0)[10]. In the Appendix 7 a detailed description of the structure of the data is given. Similarly developed code is available at git.geo.tuwien.ac.at and, if not stated differently, MIT[11] licensed. An overview of all developed and updated repositories is given in the Appendix 7.

[10] https://creativecommons.org/licenses/by/4.0/legalcode, accessed 15-August-2021
[11] https://mit-license.org/, accessed 15-August-2021

CHAPTER 3

Data Acquisition

The principal goal of this thesis is to setup and compare 2D and 3D object detection algorithm based on AR generated data. To achieve this goal, first such a comparable 2D - 3D dataset must be generated. This chapter details the creation of the new, so called *AR-2/3*, dataset. The first section describes the applied methodology, including the data acquisition concept, setup, realization, and required processing steps. The second sections presents the results both for the data acquisition itself and for the applied processing step. The third and final section evaluate AR-2/3 and positions it in respect to current 2D and 3D state-of-the-art datasets.

3.1 Methodology

3.1.1 Dataset Requirements

Already a number of 2D and 3D datasets are available, but none covers the scenario detailed below. The focus of AR-2/3 is twofold. The first open issue is the lack of a comparable, general purpose 2D / 3D datasets. Where the term *general purpose* stands for not domain specific and applies to both, presented objects and the surrounding environment. Popular 2D datasets such as PASCAL VOC or MS COCO (see Subsection 2.1.2) cover the general-purpose-requirements but obviously only contain 2D data. 3D datasets often also provide 2D data, supporting 2D / 3D comparison, but do not comply to the idea of general purpose. There are 3D datasets such as KITTI (see Subsection 2.2.2) covering the outdoor driving domain with samples of cars, pedestrians and cyclist and there are datasets such as SUN RGB-D or FAT (see Subsection 2.2.2) depicting commodity objects but only in an indoor environment. Hence, the new dataset should bridge the gap between indoor / outdoor environments, generic object categories and the possibility of 2D / 3D comparison. The second topic addressed in AR-2/3 is the absence of a AR generated datasets. Available standard datasets have two disadvantages when it

comes to applicability to the AR domain. First, the image perspective as recorded by a wearable AR device might be quite different to images available in standard dataset which are often taken by hand [Li et al., 2020]. Second, despite 3D datasets provide a number of different 3D data representations, none of them is AR generated. AR 3D data might including automatic data accumulation or internal preprocessing steps which affect the user accessible data [Hübner et al., 2020].

Based on these two foci some general requirements are derived.

1. Generic object categories must be employed.
2. Both indoor and outdoor environment must be included.
3. Both 2D and 3D data must be acquired. The same conditions must apply.
4. Data must be acquired with an AR device.

To create a meaningful generic dataset a careful selection of object categories, object instances and environment conditions is required. Based on this selection it should additionally be possible to properly evaluate AR sensing capabilities and to analyze the effect of a set of properties and conditions on detection algorithms. For this the selected object categories need to cover a set of object properties and defined intra-category variabilities. In detail considered object properties should be size, geometric complexity, and surface characteristics. These three properties include different degrees of difficulty for both data acquisition and object detection.

Object size should range from large objects where only parts are visible in some images, to quite small objects which may cover only a small part of the scene. This setup tests on one hand the sensing capability of small objects and on the other hand the used object detection algorithm is required to understand both the overall, big picture of a scene but also small, maybe non-iconic details.

Geometric complexity should have values from simple (e.g. cuboid objects) to complex. Complex objects include difficultly ascertainable (e.g. thin) structures, irregular shapes which are complicated to describe with standard geometries, and / or layovers of multiple structures. Especially geometrically complex objects offer valuable information about the sensing capabilities of the different methods - is it possible to capture thin structures or not? Geometrically simple objects can be seen as reference measurements. Next to sensing capabilities also the potential of detection algorithms to handle multiple levels of geometric complexity are covered.

Surface characteristics include two main properties: color and reflectivity. Color is especially important in the 2D image domain, as detection algorithms are solely based on RGB input [Zou et al., 2019]. Reflecting surfaces may appear differently depending one position, viewing angle, and lightning. This can affect the measured 3D data [Zeller, 2018], but also introduce variability in 2D object representations. To account for this, both highly reflective and non-reflective object categories should be included.

	properties			intra-category variability		
	object size	geom. comp.	reflectivity	object size	geom. comp.	reflectivity
chair	rather large	rather complex	rather refl.	low	low	rather low
cup	small	simple	rather refl.	low	low	low
monitor	rather small	simple	refl.	low	low	low
pottedplant	small	complex	non-refl.	rather high	rather high	low
table	large	simple	non-refl.	high	low	low

Table 3.1: Distribution of object properties and intra-category variabilities within the selected categories.

Intra-category variabilities describe effects due to diversity of the aforementioned object properties within a single category. Variabilities are possible for each object property, such as size or reflectivity. High intra-category variability may imply the requirement for more complex abstraction in the detection algorithm and therefore different variability levels can describe an algorithm's abstraction capability.

Next to object centric effects also the impact of the environmental conditions — indoor vs. outdoor — and lightning is of interest. On one hand 2D algorithms are normally evaluated on standard datasets which often do not distinguish between indoor and outdoor but include images from both environments, on the other hand 3D algorithms normally focus either on indoor or outdoor scenes. Moreover, environment and lightning can affect reflectivity which is an additional concern for 3D AR generated data. Therefore it may be more difficult for a 3D algorithm to handle both environments.

To eliminate interaction effects between categories each scene should include only a single object instance. Moreover, the direct surrounding environment should be controlled. This should ensure that no disregarded, systematic effects are introduced. Consider the following example: Large objects are placed on the ground and small ones on a platform. The platform surface is consistently flat and smooth, the ground may also be flat and smooth for the indoor conditions, but in the outdoor conditions it could be for instance grass and therefore rough and inhomogeneous. This could introduce a correlation between the large object categories and rough surfaces. Hence, to allow an easier and more exact analysis of the different characteristic and conditions these two additional restrictions are applied.

3.1.2 Dataset Concept

Based on the required properties above five object categories are selected: *chair*, *cup*, *monitor*, *pottedplant*, and *table*. Each category is composed of ten unique instances. The categories cover different values of the described object properties and intra-category variabilities as shown in Table 3.1.

To evaluate lightning and environmental effects four conditions are defined. Two indoor and two outdoor environments with four different lightning conditions are considered.

The indoor environments are controlled and static. There are no environmental changes between object instances. In contrast the outdoor environments are not completely static, small changes in the background and lightning conditions are included.

1. *indoor-night* (IN): Controlled indoor environment with constant artificial light from the ceiling. To ensure that no additional natural light is present, data is acquired on cloudy days or after sunset. Windows are additionally darkened with blinds. This setup ensures constant lightning conditions for all object instances.
2. *indoor-sun* (IS): Controlled indoor environment with natural light from windows. No artificial light is used. Therefore lightning and lightning effects such as shadows vary between object instances randomly.
3. *outdoor-night* (ON): Although the condition is called outdoor-*night* for naming consistency, it is not acquired during night time but describes the *darker* outdoor condition. It shows an outdoor environment with no direct sunlight. Objects are placed in shade. The setup ensures that all object instances are captured similarly and constantly illuminated. This shade condition is preferred over a real night condition as for the latter again constant artificial light must be used, which in turn would make the two *night* conditions very similar.
4. *outdoor-sun* (OS): Sunny outdoor environment. Direct and changing sun light introduces randomly varying illumination between object instances.

These four conditions include indoor and outdoor scenes, different lightning conditions (artificial illumination, direct and indirect sun light) and different variability in object illumination.

To capture only one object instance at a time and ensure a controlled and comparable environment all object instances are placed on top of the same platform as shown in Figure 3.1. All object instances are placed centrally and oriented into the same direction. This should eliminate directional effects related to the relative placement on the platform. A rectangular platform is used to allow observations over multiple distances. This includes observing the object instances while walking on the ground but also while walking on top of the platform. For the latter a large enough platform extend with 3.6×2.4m is selected. As the platform is present in all observations it is made as neutral as possible. It is completely covered in light brown paper. The color is neutral in RGB images and the material — non-reflecting, opaque and light — provides optimal conditions for 3D depth measurements, required for the spatial mesh generation.

In summary, the dataset consists of 2D and 3D measurements. A measurement includes only one annotated object located centrally on top of a platform. Each object belongs to one of five categories (*chair*, *cup*, *monitor*, *pottedplant*, and *table*). Each category is composed of ten instances. Each instance is observed in four different conditions (*indoor-night*, *indoor-sun*, *outdoor-night*, *outdoor-sun*). Hence, in total 50 different object instances observed in four conditions are included in the dataset.

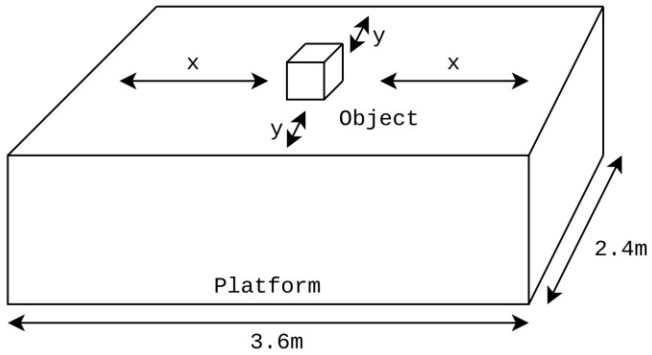


Figure 3.1: Object configuration for data acquisition: Each object instance is placed centrally on top of a 3.6×2.4m platform.

3.1.3 Technical Setup

The presented dataset concept already covers most stated requirements. To fulfill the requirement of an AR generated dataset Microsoft's AR glasses HoloLens 2 [Microsoft, 2021b] are used for data acquisition. HoloLens 2 is capable of capturing both 2D and 3D data detailed in Subsection 2.3.3. In this thesis RGB images acquired with HoloLens 2's frontal RGB camera are used. Multiple 3D data sources are available either leveraging Research Mode API or the spatial mapping capability. Research Mode offers a wide range of depth information but has the drawback of being only available when the device is in *development mode* [Ungureanu et al., 2020]. As this is an infeasible requirement Research Mode is neglected here. In contrast the spatial mapping capability is always accessible also in a production environment [Zeller, 2018]. Therefore the triangular mesh provided by the spatial mapping capability is used as 3D data source.

HoloLens 2 provides only limited storage and processing resources. In detail only 64 GB of storage are available on the device [Microsoft, 2021b]. Initial tests showed that about 300 MB of data are acquired within 100 sec. Based on this information within an hour more than 10 GB of data could be acquired and need to be stored. For data acquisition consistent and large enough storage is a central requirement. Hence, to allow continuous and stable data acquisition, data is directly transferred to an external server. With the separate server the storage could be arbitrarily increased. It should be mentioned that aforementioned advantages presume the server to run on a machine with higher specifications than HoloLens 2. Moreover, this is a useful setup for latter object detection, overcoming the restricted processing capabilities, allowing processing on GPUs and thereby enabling support of more complex algorithms (see e.g. [Eckert et al., 2018], [Farasin et al., 2020]).

In summary, processing and data management is separated into two components: server and client. A number of frameworks are available both on server and client side. Centrally an adequate web application framework has to be selected. Considered frameworks should be free of charge, well documented, actively maintained and ensure quick prototyping.

Next to server and client side frameworks a communication protocol must be selected. Here the setup leverages the internet protocol suite, composed of four layers: link, internet, transport and application layer. For the bottom link layer Ethernet is employed. Ethernet provides both lower latency and higher throughput compared to other forms of communication support by HoloLens 2 such as Bluetooth or WiFi. Addressing on the network layer is done using IPv6 link local addresses to avoid the need for a DHCP server on the same network. To have an a-priori known IP on the HoloServer EUI-64 is used to generate the IPv6 link local address based on the MAC address ot the device. TCP is used as transport layer. For the top application layer http is used due to good library support both on client and server size. This also increases the development speed compared to using plain, TCP or UDP sockets. Http comes with a small protocol overhead. As big data volumes are expected to be transferred this overhead can be neglected.

For the server implementation — called *HoloServer*[1] — the *Python*[2] programming language is selected, fulfilling above mentioned framework requirements. Additionally the author has some experience with this programming language, promising fast results. On top of Python the web framework *Flask*[3] is selected as it is lightweight and open source. The main responsibility of HoloServer is to consistently and persistently store 2D and 3D data. In detail 2D images are persisted as RGB images in PNG format and 3D scenes are stored as OBJ files. This storage capabilities are accessible via a REST API. Endpoints are tailored to data acquired by HoloLens 2 but may also be accessed by other systems providing the same data formats.

- `/`
  - *Method:* GET
  - *Parameters:* None
  - *Returns:* Health signal, whether the server is up and running
- `/store/img`
  - *Method:* POST
  - *Parameters:* Image byte stream
  - *Returns:* The filepath to the stored PNG image.
- `/store/mesh`
  - *Method:* POST
  - *Parameters:* Object byte stream
  - *Returns:* The filepath to the stored OBJ file.

To acquire data with HoloLens 2 two clients — referred to as *HoloClient*[4] — are developed, one to handle 2D data acquisition and one to handle 3D data acquisition. As clients must be running on HoloLens 2 both are implemented as *Unity*[5] scenes and use *C#*[6] for scripting.

[1] https://git.geo.tuwien.ac.at/markus-kattenbeck/holo2_server, accessed 14-April-2021
[2] https://www.python.org/, accessed 13-April-2021
[3] https://flask.palletsprojects.com/en/1.1.x/, accessed 13-April-2021
[4] https://git.geo.tuwien.ac.at/markus-kattenbeck/holo2_client, accessed 14-April-2021
[5] https://unity.com/, accessed 13-April-2021
[6] https://docs.microsoft.com/en-us/dotnet/csharp/, accessed 13-April-2021

To simplify interaction with HoloLens 2 *Microsoft's Mixed Reality Toolkit (MRTK)* 2.3[7] is employed. HTTP communication is based on `Windows.Web.Http.HttpClient`[8] leveraging that HoloLens 2 is a Universal Windows Platform device.

Microsoft also provides *HoloLenForCV*[9], a toolkit to facilitate computer vision research with HoloLens. Although this is an interesting project to retrieve and process image data it is only compatible with HoloLens first generation. The successor project *HoloLens2ForCV*[10] was not available at the time of prototyping. Therefore the 2D client is based on the currently not actively maintained, but stable *HoloLensCameraStream*[11] project. This minimalist package provides the required functionality to access HoloLens 2's RGB camera images. Camera parameters such as resolution and frame rate can be easily configured. Once setup `OnFrameSampleAcquired` method is called when a frame is ready. Listing 3.1 shows how to access the current frame and send it to the server.

Listing 3.1: HoloLensCameraStream image processing

```
1
2 public class CameraStreamRunner : MonoBehaviour
3 {
4 ...
5 private void OnFrameSampleAcquired(HoloLensCameraStream.VideoCaptureSample
      sample)
6 {
7 // prepare image bytes
8 if (_latestImageBytes == null || _latestImageBytes.Length < sample.
    dataLength)
9 _latestImageBytes = new byte[sample.dataLength];
10 sample.CopyRawImageDataIntoBuffer(_latestImageBytes);
11
12 // Send image to server
13 httpClientHandler.StoreImage(_latestImageBytes);
14 }
15 ...
16 }
```

The 3D data client uses MRTK spatial mapping observer to access the spatial mesh. Listing 3.2 summarizes the storage procedure of a spatial mesh. On access all parts of the current spatial mesh are collected and serialized. As no 3D object detection algorithm can currently handle triangular mesh data, only the points are serialized, converting the mesh into a point cloud. This has the additional advantage of decreasing the data volume of each 3D scene. Then the serialized content is send to the server. As data transfer times depend on the transmitted data volume and the size of the mesh increases with

[7] https://github.com/Microsoft/MixedRealityToolkit-Unity, accessed 13-April-2021
[8] https://docs.microsoft.com/en-us/uwp/api/windows.web.http.httpclient, accessed 14-April-2021
[9] https://github.com/Microsoft/HoloLensForCV, accessed 14-April-2021
[10] https://github.com/Microsoft/HoloLens2ForCV, accessed 14-April-2021
[11] https://github.com/VulcanTechnologies/HoloLensCameraStream, accessed 13-April-2021

longer observation times, it is decided to not use a fixed frame rate. Instead a new mesh is only accessed and send once the last request finished.

Listing 3.2: MeshHandler access and storage of mesh data

```
public class MeshHandler : MonoBehaviour
{
...
public async void SendMesh()
{
IMixedRealitySpatialAwarenessMeshObserver meshObserver = GetMeshObserver()
    ;

// get all mesh parts
var targets = new HashSet<Transform>();
foreach (SpatialAwarenessMeshObject meshObject in meshObserver.Meshes.
    Values)
{
targets.Add(meshObject.GameObject.transform.parent);
}

// Serialize mesh
foreach (var target in targets)
{
string content;
content = await ObjPointWriterUtility.CreateOBJFileContentAsync(target.
    gameObject, true);
}

// Send mesh to server
httpClientHandler.StoreMesh(content);
}
...
}
```

This setup allows a seamless data acquisition. Data is acquired by the clients running on HoloLens 2, then sent to the server via HTTP and finally stored on the server. In the following the physical setup and configuration of the data acquisition is described.

3.1.4 Configuration

Three distinct locations are selected for data acquisition. The same room is selected for both indoor conditions and the two lightning conditions are implemented as described in Subsection 3.1.2. For the two outdoor conditions two spatially close but separate location are selected to ensure long enough periods with shade as well as direct sunlight. Due to technical reasons first all indoor, then all outdoor data is collected.

The usage of two clients implies the separate acquisition of 2D and 3D data. Therefor first 2D, then 3D data is collected. But — to ensure nearly equal conditions — directly one after the other. This separation allows to account for the different data characteristics

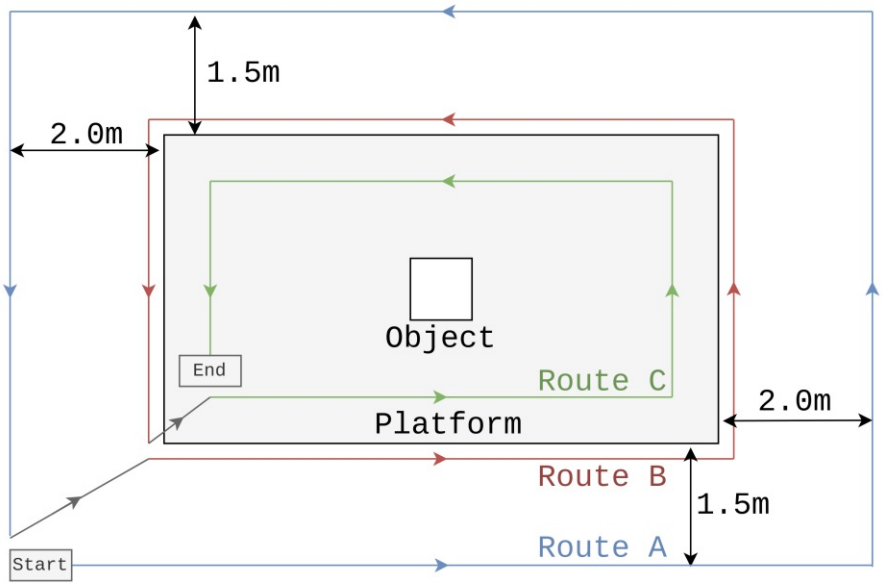


Figure 3.2: Setup configuration for 2D data acquisition: Each object instance is observed 1.5 minutes, surrounding it three times clockwise, each time focusing on a different distance and observation angle. In *Route A* the object instance is captured from far distances, therefor images generally show the whole object instance and also include the surrounding environment. In *Route B* the distance to the object is smaller and a detailed view of all sides of the object instances is acquired. In *Route C* the focus is on the top view. The observer walks — different then in Route A and B — on top of the platform.

by creating two slightly deviating setup configurations. Common to both configuration is that object instances are always placed centrally on top of the platform, oriented into the same direction and that object instances are observed with HoloLens 2 while surrounding the platform.

2D Data

One advantage of 2D data is, that it can be acquired quickly. HoloLens 2 provides RGB images at 30 fps and the transferred data volume is small especially for low resolutions. Although this high frame rate is available initial tests showed that only 7 to 8 fps give meaningful results even when using the lowest resolution of 896 × 504 pixels. This has two main reasons. First, a certain amount of time is required to send and store data on the server. Even with an ethernet connection bandwidth is limited. Second, succeeding images should still provide some differences otherwise the object detection algorithm does not benefit from a large number of samples. When walking too fast HoloLens 2's autofocus fails and images get blurry. Hence, it is required to walk slowly and in turn use a lower frame rate to capture different views of an object instance.

The requirements state that object instances must be observed from different perspectives,

this includes changing distance between HoloLens 2 and object instances and varying observation angles. To account for this, each object instances is observed from three different, clockwise running routes depicted in Figure 3.2. While walking along the routes the observer is constantly looking at the object instance. *Route A* focuses on far field observation. The distance between HoloLens 2 and object instance center ranges from 2.7 to 3.8m. In this setup object instances may only cover a small part of the images and large parts depict the surrounding environment. In *Route B* the observer walks directly along the platform. Hence, the distance to the object instance center is between 1.2 and 1.8m. The focus of this route is to capture detailed side, frontal and backward views of object instances. This also includes images showing only object parts in case of large object instances. In the last route, *Route C*, the observer walks on top of the platform, to capture the top view. Due to the small distance between object instance and HoloLens 2 — constraint by the platform extend — also observations from this route include a detailed and sometimes only partly view of object instances.

To capture approximately the same number of images per route, the same amount of time is spent on each route. Considering the risk of blurry images when walking too fast, observation time per route is fixed to 30 seconds. This sums up to a total observation time for 2D data of 1.5 minutes per object instance and condition. As the observation is a manual task these times are only target values and may vary slightly between single observations. With this target observation time and a frame rate of 7.5 fps over 600 images are acquired per object instance and condition. This results in approximately 120,000 images.

3D Data

The spatial mesh representation used as 3D representation has some fundamental different properties than 2D image data. The first important difference is that the spatial mesh can not be created instantaneously, but in a continuous and cumulative manner. First some initial parts of the mesh are calculated and then it is further extended or updated. These initial parts should already contain object instances or at least parts of them. Mesh data not containing any object instances must be discarded as there is nothing to be detected by an object detection algorithm. The second significant difference is that measured point locations are absolute. Therefore, the distance between two measured points is independent of the distance to the measurement device (HoloLens 2). But the location and orientation of the device defines for which area the spatial mesh gets generated or updated.

To generate or update the mesh in practice calm and slow movement is required. Even a walking speed as used for the 2D data acquisition is not possible. If a mesh already exists, too fast movement can lead to mapping loss events. This is a phenomenon where HoloLens 2 cannot locate itself within the previously captured mesh. The mesh looses its anchors in the real world and seems to be floating around. New measurements are added at possibly wrong locations which results in a corrupt mesh.

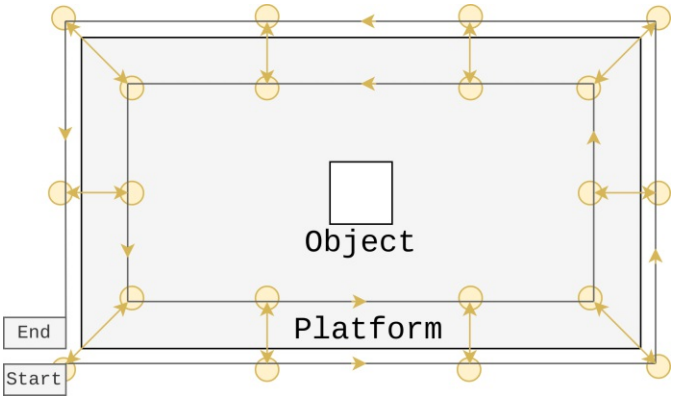


Figure 3.3: Setup configuration for 3D data acquisition: Each object instance is observed 6 minutes, surrounding it clockwise. While walking around the object instance the observer constantly changes between the frontal or side view visible from the ground to the top view perspective captured standing on top of the platform.

Based on these restrictions of HoloLens 2, 3D data acquisition firstly focuses on generating a meaningful mesh containing the object instances or parts of it. According to this main focus the 3D data acquisition route is defined as presented in Figure 3.3. Compared to 2D data acquisition it is based only on a single route depicted in yellow and gray. Before observing an object instance the old mesh is deleted to ensure independent measurements. Then observation starts at the lower left corner of the platform looking directly towards the object instances. This ensures that already the first versions of the spatial mesh includes the current object instance.

In contrast to the 2D data acquisition, the observer does not simply walk along the predefined route but rather follows a stop-and-go behavior. At the yellow circles the observer stops and looks at the object instances slightly moving its head and upper body to allow small variation in the viewing geometry and eliminate for HoloLens 2 difficult perspectives e.g. due to reflections. These movements are adopted on a per instance base to get an optimal mesh representation. To further extend HoloLens' field of view the observer constantly switches between frontal / backward / side and top view by stepping onto the platform and back down again. Some object instances may be difficult to capture from a frontal / backward / side or top view therefore these continuous changes ensures that constantly both viewing angles are available and also a number of different mesh representations of the object instance are generated.

3D data acquisition is a lot more time consuming then 2D data acquisition, on one hand because the mesh generation is more expensive and on the other hand because the next dataset is only transferred once the previous one succeeded — hence no parallel processing. Therefore an observation time of 6 minutes is selected. Within this time frame more than 300 different mesh representations can be acquired, resulting into about 60,000 3D scenes in total. The exact number per object instance strongly depends on

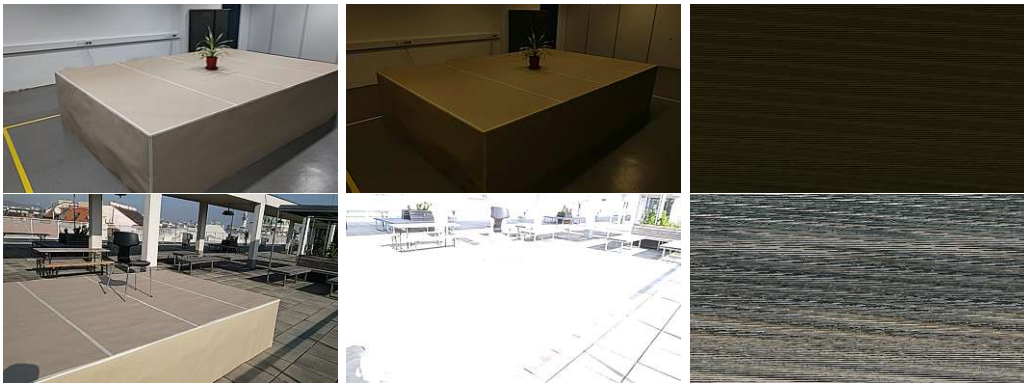


Figure 3.4: Comparison between standard images (left), ones with incorrect settings (middle) and corrupt captures (right). The latter two can be the result of the initial adjust period of HoloLens 2.

the generated data volume as the next mesh is only transferred once the predecessing request returned.

Each of the 50 object instances are observed twice in four conditions, once for 2D and once for 3D data acquisition. 2D data is acquired along 3 routes over 1.5 minutes covering different observation distances and angles. 3D data acquisition is based on a stop-and-go setup continuously integrating different viewing geometries to optimize the mesh representation of the object instance. It is with 6 minutes more time consuming. After describing data acquisition the focus now shifts towards processing the generated 2D and 3D datasets.

3.1.5 Processing

The raw data acquired with HoloLens 2 and stored at HoloServer requires some processing before meaningfully using it to train object detection algorithms. Some unsuitable scenes are still included e.g. captured during startup or cool-down. Additionally, the raw data only contains classification information — only the correct category but not the bounding box is know. Therefore object instances have to be annotated. As 2D and 3D data must be handled differently, in the following also the processing steps are detailed separately. First 2D and then 3D data preparation is described.

2D Data

In the initial data cleaning step all unplanned or unsuitable images are removed from the dataset. Such images are created either when starting or stopping the application.

When starting the 2D data acquisition client, images captured with HoloLens 2's RGB camera are directly sent to HoloServer. Experiments show that HoloLens 2 requires some milliseconds to adjust to environmental / lightning conditions. This applies to all tested

Figure 3.5: Two examples of images containing the hand gesture to stop 2D data acquisition.

conditions. During this time up to ten images are captured which are different than all other images acquired afterwards. Two examples are depicted in Figure 3.4. The upper row is an example from the *indoor-night* condition, the bottom row from the *outdoor-sun* condition. The left column shows images as acquired most of the time. The middle and right column presents images acquired during startup. To get a consistent dataset, images as in the middle and right column are manually removed.

Also stopping the 2D client creates problematic images. The client is ended by a hand gesture. While doing this hand gesture images are still captured and stored on HoloServer. Two examples are shown in Figure 3.5. As images with a hand could add additional unplanned and uncontrolled effects which are also not present in 3D data, all images containing hands are manually removed. Moreover, as data acquisition is a manual process some images exist not depicting any object. For consistency reasons also those ones are manually removed.

After data cleaning the remaining images are annotated. Different options for labeling are considered. Manually labeling the whole dataset is due to the number of images — about 120,000 — not possible. No simple algorithms solution is available and even using networks such as YOLOv3 trained on MS COCO does not achieve convincing results. Finally the idea is to label images by applying transfer learning and slight overvfitting. Hence, pre-trained weights of a neural network are used as a starting point to retrain the network. This way the network already starts of with knowledge how to solve a similar problem. The network is retrained with a manually labeled subset of the new dataset. The resulting weights can then be used to label the whole dataset. As images are quite similar — acquired with 7.5 fps while walking slowly, showing a consistent scenario — a meaningfully selected subset should be enough to cover most cases. Additionally, no data augmentation is applied so only the observed lightning conditions and perspectives are learned.

In total 6,000 images are labeled manually with Microsoft VOTT[12]. The manually labeled data is composed of 30 random images per instance and condition (see `InputData-Manager`[13]). Hence, a completely equal distribution between instance and condition

[12]https://github.com/Microsoft/VoTT, accessed 16-April-2021

[13]https://git.geo.tuwien.ac.at/markus-kattenbeck/holo2_server/-/blob/master/app/resources/yolov3/input_file_manager.py#L11, accessed 17-May-2021

is employed, while accounting for systematic effects due to perspectives by randomly selecting the image per instance and condition. The 6,000 annotated images are distributed into 5,000 training and 1,000 validation samples. To keep the equal distribution 25 images per instance and condition are used for training and 5 samples per instance and condition are used for validation.

YOLOv3 is selected as neural network, as it achieves sufficient accuracy in a reasonable amount of time and a simple and well-documented implementation is available: `yolov3-tf2`[14]. Also pre-trained weights on the current stat-of-the-art 2D dataset MS COCO are accessible. The network is trained with different hyperparameters, especially focusing onto different learning rates, batch size and initial weights. Learning rates between 0.1 and 0.00001 are considered. Batch size values range from 4 to 128. Different constellations of initial and fixed weights are tested. In detail either all weights are initialized randomly (*none*), only DarkNet weights are transferred from pre-training on MS COCO (*darknet*), all weights are transferred besides the output layer (*no_output*) or DarkNet (*darknet-freeze*) respectively all (*no_output-freeze*) weights are transferred and fixed. Additionally, both the full YOLOv3 implementation and YOLOv3-tiny are tested. All tests are implemented on up to five NVIDIA GeForce GTX 1060 based graphics cards with 6GB of RAM each.

The weights achieving the lowest AP_{2D} (also written as AP_{2D}@[.5:.05:.95]) on the validation set are then used to label the whole dataset. This measure is selected as it is the state-of-the-art comparison metric and not only considers an IoU threshold of 50% but IoU thresholds up to 95%. A good bounding box accuracy is essential, as the result of this labeling process is the input to the actual object detection experiment. In the labeling procedure[15] an additional check is included to ensure each image contains only a single annotation — as always only one object instance is captured — and the correct category is assigned. Therefore the labeling process includes another data cleaning step, removing all incorrectly labeled images. Solely the exact location of the resulting bounding box can obviously not be checked.

3D Data

After describing processing steps to clean and annotated the 2D datasets, in this part the processing of the 3D dataset is detailed. In contrast to the tedious, manual data cleaning process required for the 2D dataset, 3D data processing is based on an algorithmic approach which automatically finds non-fitting scenes such as point clouds not containing any objects or ones created after a mapping loss event. Therefore no manual data cleaning step is required.

Due to the geometrically well defined structure of each scene, 3D bounding boxes — but no category labels — can be retrieved algorithmically. As depicted in Figure 3.1 each object instances is placed on top of a large platform. The focus of 3D data acquisition is

[14] https://github.com/zzh8829/yolov3-tf2, accessed 16-April-2021
[15] https://git.geo.tuwien.ac.at/sherrmann/yolov3-tf2/-/blob/master/label.py, accessed 17-May-2021

on capturing the object instance. Hence, the observer's focus is constantly on the object instance. Thereby also the the platform is constantly observed. This means all measured 3D scenes contain the large, smooth plane of the platform top. In most cases it is even the largest connected plane in the scene, which makes it easy to detect. Based on the detection of the platform plane also the location of the object instance can be defined. According to the setup, only the wanted object instance is place above the platform plane. The algorithm leverages these two central assumptions: 1) the platform top is the largest plane and 2) the object instance is the only object above the platform plane.

For implementation again the Python programming language is selected, due to the author's experience with it. Moreover, the Python library *Open3D*[16] provides simple and intuitive functionalities for point cloud processing. Leveraging Open3D bounding boxes can be retrieved as presented in Listing 3.3. First, the platform is retrieved from the input point cloud. For this RANSAC [Fischler and Bolles, 1981] is used to segment the plane containing most points. Then all points located above the platform — possible object points — are extracted. In this step outliers are removed. A given number of points must be located within a given radius from a point, so it is kept. After this step only point clouds with a minimum number of object points are considered. x and y extent is defined by these object points. For the z extent the highest object point and the lowest platform point is used. The latter ensures that each object is standing on something and not floating in free space.

Listing 3.3: Creation of a 3D bounding box based on an input point cloud

```
1 class PointCloudBboxHandler:
2 ...
3 def get_object_with_below_platform_bbox(self, pcd: PointCloud) -> Optional
      [OrientedBoundingBox]:
4 # Get platform
5 platform_bbox = self.get_platform_top_bbox(pcd)
6
7 # Get and check object points
8 object_pcd = self.get_object_points(pcd, platform_bbox)
9 object_points = np.asarray(object_pcd.points)
10 num_points = object_points.shape[0]
11 if 0 < num_points < self.min_num_object_points:
12 return self.handle_not_enough_points(object_pcd)
13
14 # Get z extend
15 zmax = np.max(object_points, axis=0)[2]
16 _, _, zmin = platform_bbox.get_min_bound()
17 return self._get_oriented_bbox_from_points(object_pcd, zmin, zmax)
18 ...
```

To further improve the bounding box retrieval the a-priori known category labels are integrated into the procedure. Different geometric properties of categories are accounted for by leveraging — or not leveraging — some additional functionalities accessible by

[16] http://www.open3d.org/, accessed 16-April-2021

parameters. Due to the category based configuration a factory class is defined. Listing 3.4 provides insights into required configurations per category. Given parameters have the following effects: The considered object point space above the retrieved platform can be restricted with `relevant_platform_percentage` (in the x,y-plane) and `max_obj_height` (in z-direction). These parameters are useful to automatically remove most outliers centered above the object or along the edges of the platform. To also check the plane with the second largest amount of points `check_sec_plane` can be used. In special cases outlier removal can be skipped with `outlier_criteria` and the RANSAC parameter can be adopted with `ransac_distance_threshold`.

Listing 3.4: Used PointCloudBboxHandler configurations per category

```
class PointCloudBBoxProvider:

def get(self, class_name: str) -> Optional[PointCloudBboxHandler]:
"""Return a configured PointCloudBboxHandler."""
if class_name == "chair":
return PointCloudBboxHandler(
relevant_platform_percentage=0.75,
max_obj_height=1.5,  # chair height << 1.5m
)
if class_name == "pottedplant":
return PointCloudBboxHandler(
relevant_platform_percentage=0.75,
max_obj_height=1.0,  # plant height << 1m
)
if class_name == "monitor":
return PointCloudBboxHandler(
relevant_platform_percentage=0.75,
max_obj_height=0.5,  # monitor height << 50cm
)
if class_name == "table":
return PointCloudBboxHandler(
# the table top may contain more points then the platform top
# plane with second most point must be checked too
check_sec_plane="lower_bigger",
relevant_platform_percentage=0.85,
max_obj_height=1.0,  # table height << 1m
)
if class_name == "cup":
return PointCloudBboxHandler(
# Object is so small - could be identified as outliers
outlier_criteria=None,
max_obj_height=0.2,  # cup height < 20cm
# with higher value most of the object is identified as platform
ransac_distance_threshold=0.02,
)
print(f"The class_name {class_name} is not supported!")
return None
```

To automatically check the derived bounding boxes a volume analysis is performed. As

3D measurements are absolute ones, the volume of a single instance in a given condition should be stable over all observations. A large deviation in the bounding box volume indicates that an incorrect bounding box is derived or the point cloud is corrupt. In any case that sample should not be used. In practice volumes of the derived bounding boxes are calculated and compared per instance and condition. For simplicity outliers detection is performed based on box plot analysis introduced by [McGill et al., 1978]. Outliers are defined based on the quartil Q_1 and Q_3 and the interquartile range $IQR = Q_3 - Q_1$. Values bigger than $Q_3 + 1.5IQR$ or smaller than $Q_1 - 1.5IQR$ are discarded as outliers. Samples which bounding box volumes are within the defined range are assumed to be correct combinations of a valid point cloud and a bounding box locating the object instance within it.

3.2 Results

After the detailed description of the applied data acquisition and processing methodology, this section focuses on the results. First, observations made during data acquisition and its results are described, then, the processed 2D and 3D data is presented.

3.2.1 Acquisition

Data acquisition worked smoothly both in the indoor and outdoor environments. With the setup detailed in Subsection 3.1.4 it took between 4 and 6 days per condition, resulting in a total of 15 days — on 5 days two distinct conditions were observed consecutively.

Each acquired 2D image has an approximate size of 500 to 600kB depending on the compressibility of the content. Transfer and storage procedure at HoloServer took on average 0.2214s with a standard deviation of 0.2016s. Acquired 3D data has an average data volume of approximately 45.5kB but with a high standard deviation of nearly 90.0kB. These large deviation in 3D data size arise because of the growing mesh from the beginning of the observation to its end. Figure 3.6 lower part depicts the increasing data volume as well as the sharp drop in size when the mesh is cleaned and the spatial mapping is started from scratch. A comparison between the upper and lower graphic shows that the varying data volume is also correlated with changing transfer times. For 3D data transfer and storage on average 4.005ms with standard deviation of 3.695ms are required.

In total more than 150,000 images and nearly 75,000 point clouds are acquired. As expected approximately twice as much images as point clouds are collected. Figure 3.7 details the distribution between categories and conditions. The largest number of images — with over 8,000 — are observed in the *outdoor-night* condition for categories *monitor*, *pottedplant*, and *table*, and in the *outdoor-sun* condition for category *monitor*. The least images about 7,000 or less, are acquired for category *chair* in all four conditions and for category *table* in both indoor conditions. Most point clouds — in detail over 4,000 — are collected for condition *indoor-sun* and category *chair*, and for condition *outdoor-sun* and

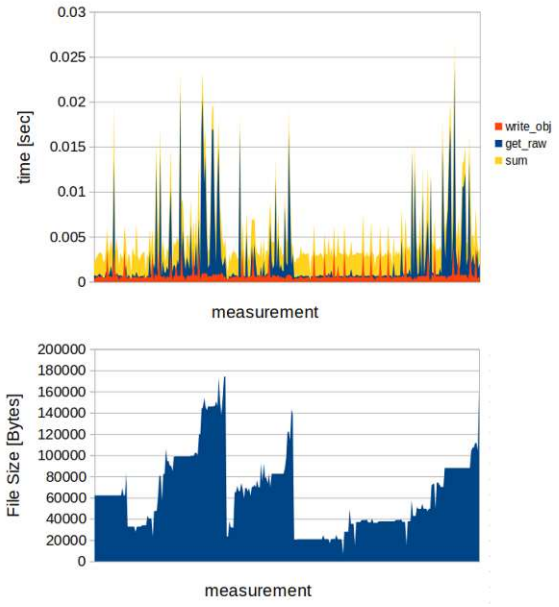


Figure 3.6: Store point cloud process example from 2020-09-14: The upper graph shows the required time for retrieving, writing and completing the whole process. The lower graph shows the data volume of the corresponding point cloud. It can be observed that the transfer time increases with the data volume and is — especially for large point clouds — the limiting factor.

category *table*. Less than 3,500 point clouds are observed in condition *indoor-night* for categories *monitor* and *pottedplant*, and in condition *outdoor-sun* for categories *cup* and *pottedplant*. Besides the slightly low number of acquired images of *chair*s no systematic effects are apparent.

Examples for each category and conditions are presented in Figure 3.8. Image and corresponding point cloud depict the same object instance captured in the same condition, solely the perspective may differ. The condition can best be differentiated from the left image column. From top to bottom the following conditions are shown: *indoor-night*, *outdoor-sun*, *outdoor-night*, *outdoor-sun*, *indoor-sun*. As intended the illumination appears constant in the *indoor-night* and *outdoor-night* condition. Particularly large changes in illumination and therefore variations in object shade and shadows created by the observer can be observed in the *outdoor-sun* condition.

Next to lightning effects also the planned object properties size, geometric complexity and surface properties and their intra-category variabilities could be captured as planned. For instance object size ranges for images from the small *cup* depicted in the second row, to the only partly and images filling depicted *table* in the last row. These size variations are also evident in the 3D representation. The cup is only described by less than ten points while most points of the last point cloud are on the table surface. A visualized example

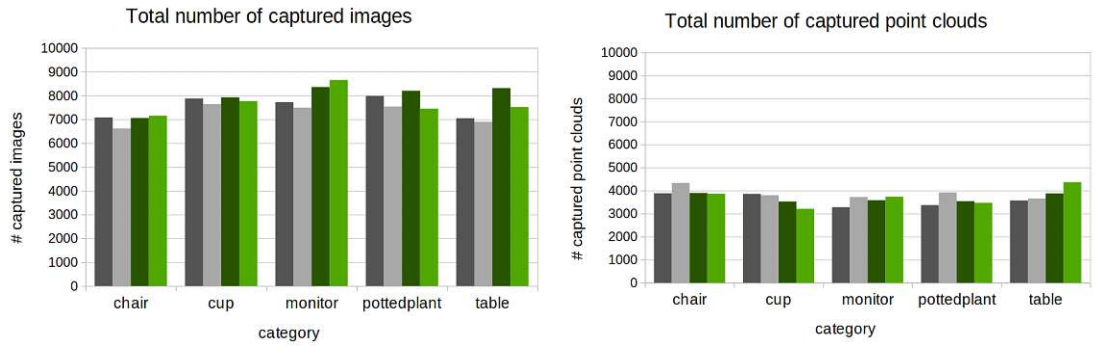


Figure 3.7: Total number of captured images (left) and point clouds (right) per condition and category. Conditions are symbolized by color: indoor-night (dark gray), indoor-sun (light gray), outdoor-night (dark green), and outdoor-sun (light green).

of a geometric complex structure is the *pottedplant* in the second last row. As shown in the images on the left side the leaves are thin and long, but HoloLens 2 captured a rougher and less structured representation. Also the geometrically simple object *monitor* is rarely captured perfectly. Most problematic are sharp and accurate edges, as this requires point measurements directly on the edge. Geometric complex parts which also have a reflective surfaces are especially difficult for HoloLens 2. For instance looking at the top row, points on the thin and shiny chair legs could not be acquired. In contrast object instance with non reflecting surfaces such as office chairs covered with fabric are particularly simple for HoloLens 2. The mesh generation works faster and more accurate even compared to slightly reflecting instances such as wooden chairs.

Demanding object size and lightning conditions also led to mapping loss events during 3D data acquisition. An example is presented in Figure 3.9, showing the correct mesh in green and the result of a mapping loss event in red. As it can be seen previously mapped surfaces are kept — though being incorrectly located — and new content is added. To ensure a meaningful and consistent spatial mesh, the acquisition is interrupted on mapping loss, the current mesh deleted, and acquisition is restarted from the origin corner. It is observed that most mapping losses happened after passing the second corner and looking into the direction of the starting point. This occurred equally in all conditions. Possibly, this happened because the observation angle changed for the first time by 90° around the z-axis and some previously visible orientation points could no longer been observed by HoloLens 2. The *outdoor-sun* condition is additionally difficult, in particular in combination with the smallest category - *cup*. An explanation could be the demanding lightning condition, due to direct sunlight and changing shadows of the observer. Moreover, it could be especially difficult for HoloLens 2 to anchor the spatial mesh — mainly consisting of the flat platform top — onto a single, small geometric structure such as a *cup*.

[17] https://www.meshlab.net/, accessed 20-April-2021

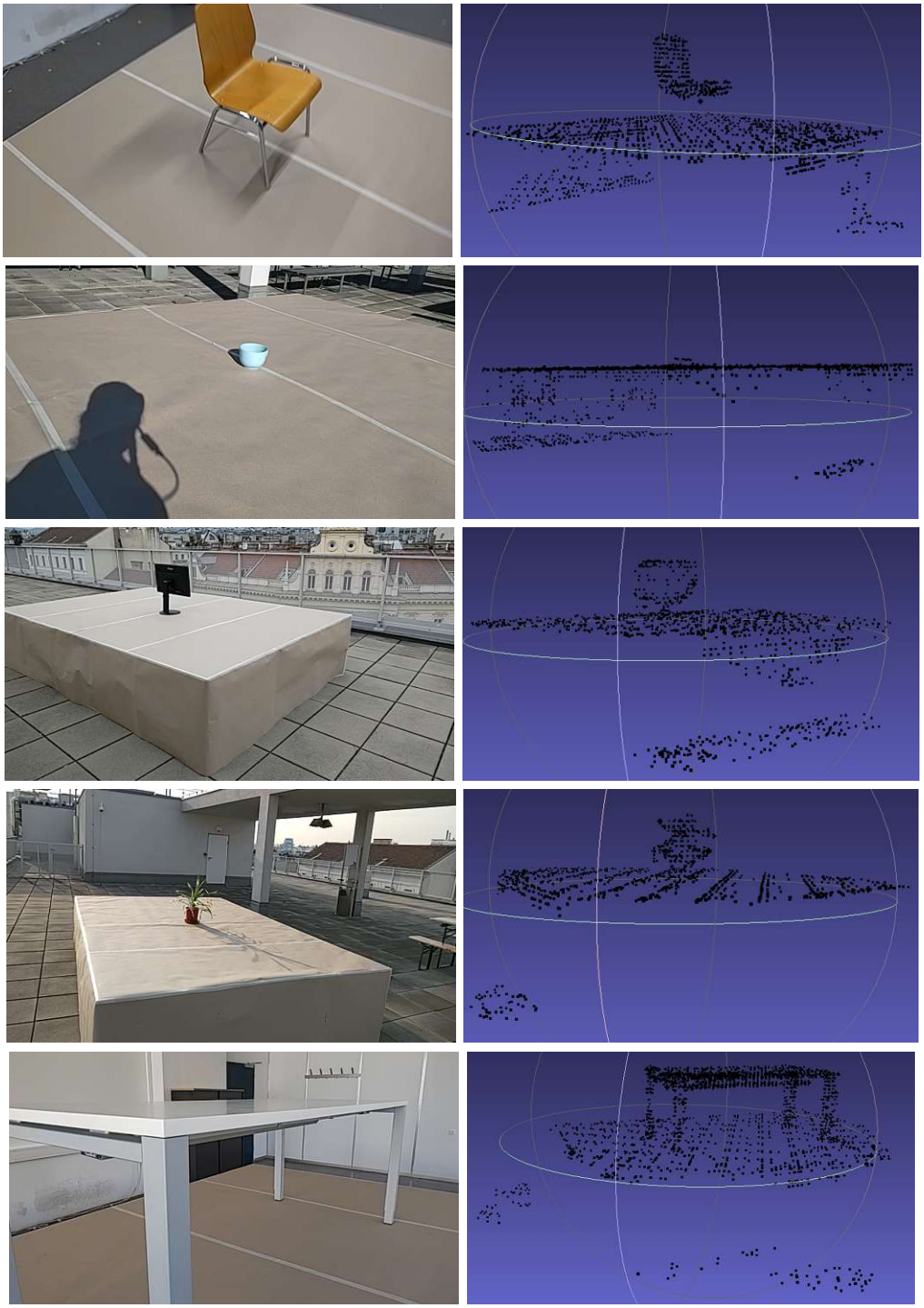


Figure 3.8: One example image and point cloud per category depicted in different conditions and from different perspectives. (Image and point cloud condition and instance match but perspective may be different.) From top to bottom: 1) chair, indoor-night, 2) cup, outdoor-sun, 3) monitor, outdoor-night, 4) pottedplant, outdoor-sun, 5) table, indoor-sun. Point cloud images use MeshLab[17]for visualization.

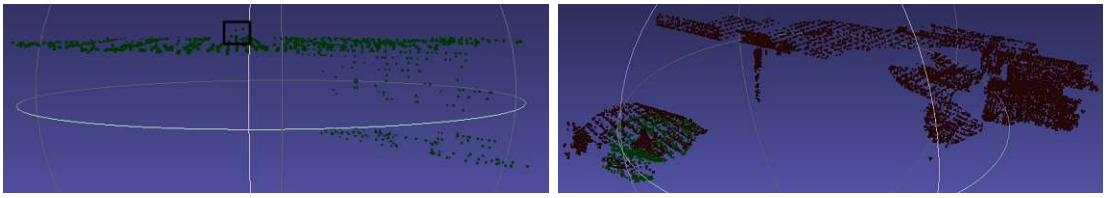


Figure 3.9: Result of a mapping loss event. The green points visualize a correct mesh representation, the red ones the point cloud after a mapping loss. In the left graphic the large horizontal and vertical surface depicts the platform, points within the black rectangle are object points — in this case a cup. Visualizations are created with Mesh

Figure 3.10: Three examples for blurry images. The left two are acquired in the *indoor-night*, and the right one in the *indoor-sun* condition.

Object properties did not have any obvious effect onto the acquired 2D data, but the different conditions did. In both indoor environment the number of blurry images is higher than in the two outdoor conditions, despite the slow walking behavior in all conditions. In detail approximately half of the *indoor-night* and a few *indoor-sun* images are at least partly blurry. Three examples are shown in Figure 3.10.

3.2.2 Processing

2D Data

After the manual data cleaning 143,587 images remain. Less then 6% — in detail 8,731 — of the acquired images are removed. 6,000 of these images are manually labeled and used to train YOLOv3, split into 5,000 train and 1,000 validation samples. To reach a good accuracy a number of different hyperparameter combinations are tested. A selection, sorted by network configuration and AP, is presented in Table 3.2, the given accuracy measures corresponds to the results after 100 epochs. In particular both the full YOLOv3 network and also its smaller tiny-YOLOv3 variation are considered. In general a comparable accuracy level is reached.

Initializing, but not freezing, weights works best in both configurations. Transferring only DarkNet weights or using all weights but the ones from the output layer does not seem to have a large effect on the final accuracy (see run 18 - 19 and run 29 - 32). The best runs starting from random weights (run 20 and 27) reached a lower accuracy. Freezing transferred weights makes the network less flexible and learning stagnates quickly.

Run	config.	transfer	learning rate	batch size	AP	AP_{50}	AP_{75}
18	**full**	**darknet**	**0.001**	**20**	**68.56**	**96.54**	**85.55**
16	full	no_output	0.001	20	66.93	97.22	82.89
20	full	none	0.001	20	57.61	91.41	67.47
12	full	no_output-freeze	0.01	64	57.21	89.09	68.28
10	full	darknet-freeze	0.01	32	51.21	82.91	55.49
4	full	darknet-freeze	0.01	64	49.55	79.56	56.78
5	full	darknet-freeze	0.0001	64	45.85	75.94	50.37
15	full	darknet	0.01	20	35.19	70.43	29.35
29	tiny	no_output	0.001	32	62.03	96.37	72.62
32	tiny	darknet	0.0001	4	60.67	96.43	71.57
28	tiny	darknet	0.001	32	59.88	96.44	69.22
27	tiny	none	0.001	4	55.87	93.38	62.49
33	tiny	no_output	0.0001	4	54.39	92.34	58.69
30	tiny	darknet-freeze	0.001	128	40.37	80.02	32.28
24	tiny	none	0.1	32	40.07	74.73	38.72
26	tiny	none	0.00001	32	14.51	38.24	7.05

Table 3.2: A comparison between hyperparameter sets training YOLOv3 with manually labeled subset. Data is sorted by network configuration and AP accuracy in descending order. The accuracy values refer to the results after 100 epochs.

Best learning rate values are between 0.01 and 0.0001 for both network configurations, higher values overshoot the minimum and oscillate around it (e.g. run 24) and with lower values progress is made only very slowly and only an insufficient accuracy is reached (e.g. run 26). In both configurations the best results could be reached with a learning rate of 0.001 (see run 18 and 29).

Also a number of batch size values are tested. Due to the high number of trainable parameters full YOLOv3 could only be trained with a maximum batch size of 4 when not freezing any weights. Using 5 machines in parallel batch size increased to 4*5=20. The lower number of trainable parameters of tiny-YOLOv3 allowed to test a higher number of batch sizes in a short period of time.

Optimal transfer strategy, learning rate and batch size are strongly connected. Results with frozen weights on the full network can be improved by using higher learning rates with smaller a batch size (see run 10, 4, and 5). The learning rate - batch size relation gets also clear comparing run 32 and 28. Similar accuracy values can be reached when increasing both learning rate and batch size. But this does not automatically apply to all hyperparameter combinations (see run 29 and 33).

The best hyperparameter combination is implemented in run 18 — full YOLOv3, initializing DarkNet weights from MS COCO, with a learning rate of 0.001 and a batch size

(a) pottedplant (b) cup (c) chair (d) table

Figure 3.11: Four image labeling examples: The left two bounding boxes are identified perfectly. The right two bounding boxes fit less perfectly.

of 20 — and achieves an Average Precision AP of 68.56%. Additional accuracy metrics calculated from the manually labeled validation set are given in Table 3.3 and 3.4.

Run	AP	AP_S	AP_M	AP_L	AR_1	AR_S	AR_M	AR_L
18	68.56	47.37	64.86	71.27	73.99	53.52	69.88	76.17

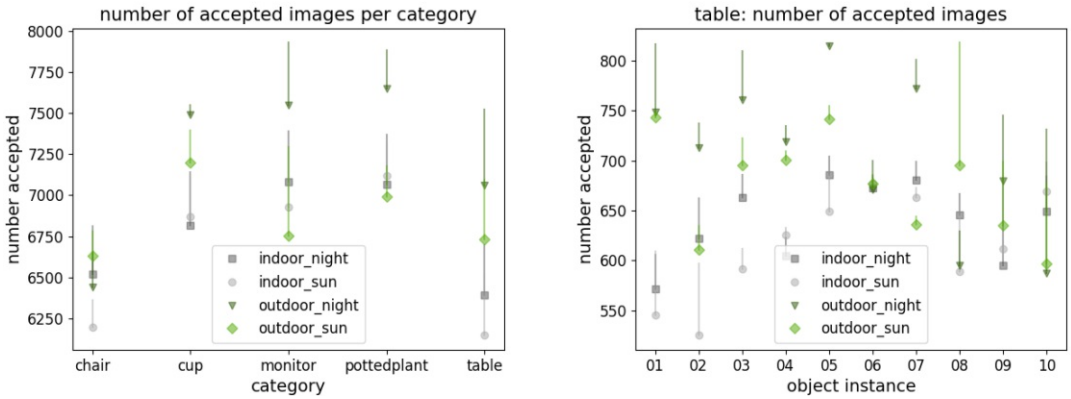
Table 3.3: MS COCO validation accuracy of best label training. AR_{10} and AR_{100} are omitted as only one object per image is present and therefore they are identical to AR_1.

Run	mAP	chair	cup	monitor	pottedplant	table
18	96.54	98.44	97.97	99.09	93.50	95.50

Table 3.4: PASCAL validation accuracy (AP_{50}) per category and overall of best label training.

Resulting weights from this training are then used to label the complete dataset. Four examples are shown in Figure 3.11. In all four the category is predicted correctly, only the bounding box fits the object better in the left two than in the right two images. In nearly all images only one object is detected, and the predictions and bounding boxes match well even for such low scores as 0.3 as for example in Figure 3.11b. Therefore all results with a score above 0.2 are accepted as long as the category is correct. But, the median score per category and condition is always between 0.8 and 1.0. In the rare case of two detections such as in Figure 3.11d only the detection with the higher score is considered, again only if the category is correct.

Figure 3.12a shows the final number of accepted image-annotation combinations. The *chair* and partly *table* category consists of the least number of images per condition. Images captured in the *outdoor-night* condition provide the most number of images for almost all categories. Looking at the number of rejected images — visualized by the bar length — no systematic effects comparing categories or conditions can be observed. For instance, although a comparable number of *table* images is rejected per condition, Figure 3.12b shows that these rejected images per condition are randomly distributed over the single instances. Over 100 *table* image from instance *08* in the *outdoor-sun* condition are



(a) Per category (b) Per instance for category table

Figure 3.12: Total number of accepted annotated images. The bar symbolizes the number of rejected image-annotation combinations, e.g. for category *monitor* condition *outdoor-night* nearly 8,000 images are acquired — top of bar — a bit less than 500 images are rejected and approximately 7,500 images are accepted — main symbol and bottom of bar.

rejected, while nearly all are kept from the *indoor-night* condition. In contrast hardly any *table* images of instance *04* are rejected.

In total 140,514 annotated images are accepted. These images are separated into train, validation, and test set. To ensure an equal distribution of categories and conditions in the validation and test set, they are each composed of 70 images per instance in a given condition (e.g. each includes 70 different images from *chair 01* in condition *indoor-night*). All remaining samples are grouped into the train dataset. Table 3.5 summarized the final split. As each images contains exactly one object the number of images and objects is identical.

# Classes	train		validation		test	
	images	objects	images	objects	images	objects
5	112,514	112,514	14,000	14,000	14,000	14,000

Table 3.5: Properties of new 2D AR-2/3 dataset.

After detailing the results of 2D data processing and considering the final cleaned and annotated dataset the focus is now shifted towards the results 3D data processing.

3D Data

With the procedure presented in Section 3.1.5 it is possible to algorithmically derive 3D bounding boxes from the acquired point clouds. Two examples are depicted in Figure

Figure 3.13: Two correctly detected objects, identified object points are colored in magenta, similar to the derived bounding box, remaining points are colored dark gray. Due to category based algorithm fine tuning correct results could be achieved also for demanding examples such as detecting the small cup the right point cloud (Image is zoomed to object points for better readability).

3.13. The left graphic visualizes the perfect case, the platform top is the plane with the highest number of points, a large number of points describe the object instance and no points beside object points are located above the platform. The right example shows a more complex scenario, only five points describe the object instance, but a number of additional, partly erroneous points are positioned above the platform. For such cases, object size restrictions per category and outlier detection are important. Both examples show correctly derived bounding boxes.

To account for somehow incorrect bounding boxes and erroneous or still incomplete point clouds the subsequent volume analysis is performed. Figure 3.14 shows two examples. The left graphic depicts the derived *chair* bounding box of a point cloud after a mapping loss event and the right graphic presents a *table* bounding box of this hardly mapped object instance. When comparing such volumes to all other volumes acquired for this object instance and condition incorrect ones can be easily identified. An example of the effect of the volume analysis is given in Figure 3.15. While all volumes vary between 0.05 and $0.45m^3$ after the volume analysis only the ones between 0.11 and $0.15m^3$ are accepted. This volume range appears logic for a potted plant.

Figure 3.16 shows the results of such an analysis by example of the category *pottedplant*. Figure 3.16a presents the total number of correctly annotated point clouds. The in color matching bar symbolizes the number of rejected point clouds. Figure3.16c gives an example how the volume distribution of a single instance and condition affects the number of not accepted point clouds. It depicts box plots per condition for *pottedplant* instance *03*. Almost all point clouds of the *outdoor-sun* condition are accepted, as volumes are quite distributed from less than 0.1 to over $0.3m^3$. Due to this large distribution quartil Q_1 and Q_3 and in turn the interquartil range IQR are large. Hence, most volumes are within the given range, even though single volumes may vary. In contrast, volumes in condition *indoor-sun* are extremely similar, this causes strict constraints — or in

Figure 3.14: Two examples of derived bounding boxes not passing the volume selection. Points within the bounding box are highlighted in pink. The left graphic shows a *chair* point cloud after an mapping loss event, the right graphic one of the initial observations of a *table*.

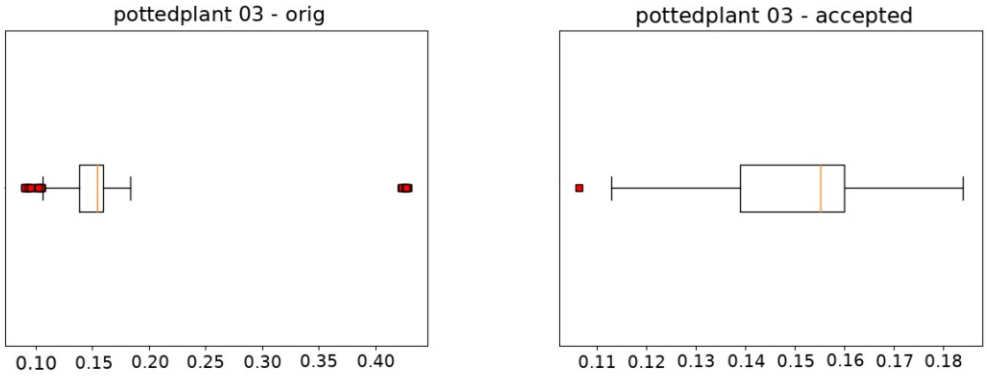
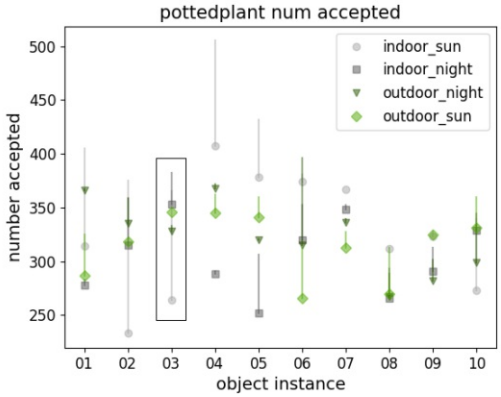


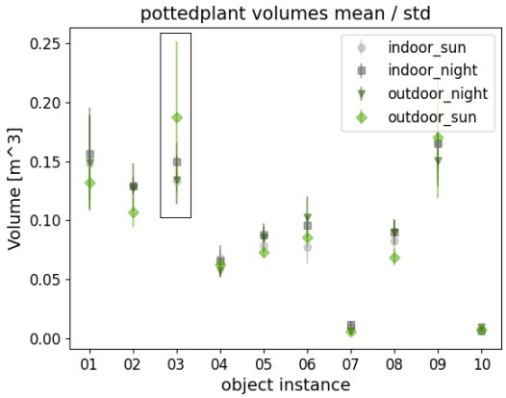
Figure 3.15: Effect of volume analysis on the example of *pottedplant* instance *04 indoor-night*. The left graphic shows the original volume box plot including volumes from 0.05 to $0.45m^3$. The right graphic show the box plot of accepted volumes, only ranging from 0.11 to $0.15m^3$.

other words all accepted volumes must be very similar — and many point clouds are discarded. Additionally Figure 3.16b and 3.16c show that the mean (and median) volume per instance is similar between the four conditions, standard deviation can differ. But, no systematic effects are detected.

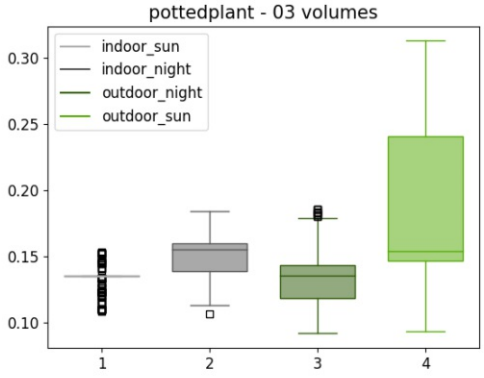
After applying the volume constraint in total 64,862 annotated point clouds are accepted. On average 300 point clouds are available per instance and condition. Point clouds are separated into a train, validation, and test dataset. As in the 2D domain validation and test set contain equally distributed samples and the train dataset summarizes all remaining ones. The dataset properties are summarized in Table 3.6. Again the number of available point clouds and objects is equal.



(a) Number of accepted pottedplant point clouds per condition and instance. The symbol defines the final number, the corresponding bar above the number of rejected point clouds.



(b) Mean (symbol) and standard deviation (bar).



(c) Volume box plot pottedplant 03.

Figure 3.16: Volume selection properties and results: The box plot and number of accepted point clouds describe together the relation between the volume distribution and the number of rejected point clouds. Both mean/std and box plot show matching volumes between conditions.

Scene type	Sources	# Classes	train	validation	test
Indoor / Outdoor	Point Cloud	5	50,862	7,000	7,000

Table 3.6: Properties of new 3D AR-2/3 dataset.

3.3 Evaluation

The previous section presents the results of the data acquisition and data processing steps of the two main parts — 2D / 3D — of the dataset separately. Also the evaluation starts by discussing properties of the 2D respectively 3D part of the dataset separately. The main focus is to position the newly created, AR-2/3 dataset in respect to existing standard datasets. After this domain based evaluation the two parts are analyzed as a whole and 2D - 3D feature with their possibilities and restrictions are detailed.

3.3.1 2D Dataset Comparison

For 2D generic object detection the most important standard datasets are presented in Subsection 2.1.2 with their properties detailed in Table 2.1. The properties of the 2D part of AR-2/3 are summarized in Table 3.5. It should be mentioned that most datasets focus on specific aspects of object detection. The two PASCAL VOC datasets are only of interest because of their historic importance. ILSVRC provides a very large number of fine-grain categories, OID makes available an even larger number of images, annotated object and categories. This allows to analyze the capability of object detectors to distinguish very similar object categories. MS COCO is composed of real world examples, providing realistic context and annotations for non-iconic objects. Because of this MS COCO is the most challenging but maybe also most realistic dataset today.

AR-2/3 has a lot less categories, in detail only 5 in comparison to 80 for MS COCO, 200 for ILSVRC and even 600 for OID. Even the nowadays outdated PASCAL VOC dataset includes 20 different categories. In contrast, the number of images is with over 140k approximately the same as for MS COCO, but is again significantly less than for ILSVRC and OID, and significantly higher than for the historic PASCAL VOC datasets. Nevertheless the number of annotated objects is considerably less than for the three current datasets. This is mainly attributed to the fact that in AR-2/3 exactly one object is annotated per image. In contrast MS COCO has on average more than 7 annotated object per images. ILSVRC is more similar to AR-2/3 with a bit more than one annotated object per category in the training dataset but about 2.7 annotated objects in the validation dataset. Due to the small number of categories the average number of annotated object per category is with more than 28,000 twice as large as for MS COCO. Hence, due to the relation of the number of images, number of annotated object, and number of categories AR-2/3 provides more samples per category than current standard datasets. Additionally, the object detector has to differentiate less categories. These properties should simplify learning and improve training results of neuronal networks.

(a) indoor-night (b) indoor-sun (c) outdoor-night (d) outdoor-sun

Figure 3.17: Instance *chair 04* from a similar perspective in the four conditions.

The used data acquisition configuration further simplifies object detection on AR-2/3. As described in Subsection 3.1.4 the same setup is used for all object instances and environment / lightning conditions. In detail an object instance is position on top of a platform and surveyed from different perspectives, following a predefined route. Due to using a frame rate of 7.5 fps and the requirement of moving slowly to prevent blurry images, consecutive images are extremely similar. Moreover, the same object instance is observed from the same route four times — once per condition. Figure 3.17 shows an examples of the category *chair* instance *04*. Even though generally in the two sun conditions some variations in the direction of the light and consequently in the shade are possible, brightness and contrast are constant within a condition for a given object instance. This means that each object instance is very well captured — from a number of different perspective and in different, static environments. But this also implies that for most acquisitions (images) there are a number of very similar ones available — images captured directly before or after a given acquisition and images captured under a different condition but from the same or at least a very similar perspective (as visualized in Figure 3.17). Assuming that only images acquired throughout this campaign make up the population, it is extremely homogeneous. Due to the structured distribution between train, validation, and test split, and the large share of about 80 % of the train split, it is in addition very likely that the train split contains a number of very similar samples for each sample in the validation respectively test split.

The setup of measuring only a single object positioned on top of a platform has additional implications. The main one is, mostly images with one, iconic object are acquired. According to [Lin et al., 2014] (creators of MS COCO) such iconic objects are less frequent in real world samples — therefore in contrast to its predecessor ILSVRC, MS COCO focuses on non-iconic and small objects (see Figure 2.2 for an example). As accuracy results suggest such non-iconic objects are more difficult to detect, or, seeing it from the opposite side, iconic objects are easier to detect. Hence, this is another indication that detection accuracy results should be higher on AR-2/3 than on MS COCO.

As described before, data is acquired in a controlled environment to enable a meaningful comparison between the different categories and environment / lightning conditions. One important difference especially to the real world focused MS COCO dataset is that objects are not captured in context, but rather isolated. Therefore object detectors must use object properties only to identify the different categories. This allows interesting conclusions about the importance of context information for specific categories. Namely,

is the context information equally important for all categories, or is it maybe more important for e.g. small categories? Moreover, the controlled environment ensures that approximately the same number of acquisitions are available per category and even per object instance and condition. Using web crawled data this property is difficult to achieve. Summarizing the implications of the data acquisition configuration it can be stated that the dataset does not describe real world situations, but is acquired under laboratory like conditions. This enables a meaningful comparison between categories, conditions and finally between 2D and 3D results.

As mentioned before no systematic differences between the number of images respectively annotated objects per category could be observed (see Figure 3.12). Hence, category and condition specific differences cannot be explained by a varying number of training samples for the respective category / condition. Although not considered as systematic effect, it should be mentioned, that the least images (25,786) are available for category *chair*. More interesting is the fact that approximately every second image in the *indoor-night* condition and some images in the *indoor-sun* condition are blurry (see e.g. Figure 3.17a). In contrast nearly all images in the two outdoor conditions are perfectly sharp. This could be an effect of the used artificial light in the *indoor-night* condition. As data for the two indoor conditions is collected first this effect may also be attributed to the increasing experience of the observer. Interestingly HoloLens 2 does not have any problems capturing images in the outdoor environment even with direct sunlight. All auto focus, auto white balance and auto exposure work fine even in this more demanding condition.

3.3.2 3D Dataset Comparison

In the 3D domain currently datasets are mainly developed for specific applications with a strong separation into indoor and outdoor (mainly autonomous driving) datasets. To the author's best knowledge, no real generic dataset exists at the moment. 2D standrad datasets differ in the sense of focusing on some general properties such as better fain-grain distinction between a large number of categories or provide real world samples with non-iconic objects. 3D datasets rather provide samples for one specific application. For this application then in turn maybe multiple datasets exist to cover the different general foci. Most important 3D datasets are presented in Subsection 2.2.2 and the datasets specifications are summarized in Table 2.4. The specification for the newly created, AR-2/3 3D dataset are listed in Table 3.6.

Comparing the two tables one observes that AR-2/3 has the lowest number of categories. But, it should be considered that KITTI is often evaluated only on its three main categories (car, pedestrian, and cyclist) and similarly for most evaluations on SUN RGB-D only the 10-class version is used. In this sense five can be seen as acceptable number of categories in comparison to current 3D standard datasets. AR-2/3 provides a bit more 3D scenes as the large, synthetically generated FAT dataset. It is significantly larger than SUN RGB-D and KITTI, but not as large as nuScenes. Similarly to the 2D domain this is not valid for the number of annotated objects as also in the 3D domain each object is

captured in isolation. With on average more than 6 annotated objects per scene SUN RGB-D provides more than 64k annotations and KITTI includes with approximately 5.3 annotations per scene more than 80k annotations. Therefore the total number of annotated objects is similar in KITTI, SUN RGB-D, and AR-2/3. In summary, AR-2/3 has similar specifications — number of categories and number of annotated objects — as the current standard dataset KITTI and SUN RGB-D.

An interesting feature of AR-2/3 is that it already provides a meaningful split into train, validation, and test split. This complete split is only provided by SUN RGB-D. The others either only split into train and test, or provide only a single dataset, to be split by the user. Additionally, AR-2/3 allows a fair comparison between the different categories as approximately the same number of annotations are provided per category. Both SUN RGB-D and KITTI have the disadvantage that for some categories a significantly larger amount of annotations are available than for others. For instance the KITTI dataset is composed of 75% car and only 4% cyclist annotations. This creates an imbalance and hinders a meaningful accuracy comparison between the different categories.

Also for the 3D data the controlled data acquisitions implies a homogeneous dataset. Although 3D data is captured less frequently than 2D data, consecutive scenes are still similar. There are two main reasons for this. First, the spatial mesh is build up continuously. In each step parts are added or updates. This is possible because 3D scenes contain absolute distance measurements, therefore also the position of each measured point is absolute and does not change in respect to the viewing angle. Second, creating the 3D spatial mesh is more complex and thereby time consuming than capturing the current RGB view of the camera. Although larger time ranger are between two consecutive acquisitions, the changes in the spatial mesh may be small. In fact because of the absolute measurements the 3D data may be even more homogeneous than the 2D data. There are no different perspectives for a given object. Observing an object from different perspectives may only lead to some additional points or an update of existing point locations. That is why largest difference between acquisitions of the same object instance can be attributed to not yet full captured objects.

The dataset uniformity is further ensured by the data acquisition setup. To best capture the currently observed object instance even from the beginning and in different variations, one requirement for the data acquisition is that the observer constantly focuses on the object. In combination with the well defined data acquisition configuration of always placing exactly one object on top of the platform, most acquired point clouds contain a limited number of points and only depict the object in an iconic manner on top of the platform. Hardly any environment is captured. Beside indirect effects of the measured point locations, acquisitions of a single object instance result into similar point clouds for the different environment / lightning conditions. This uniformity should make it easy to learn the different categories by a neural network. Hence, it is expected that better object detection results can be achieved than for other datasets with comparable specifications.

In an analogous manner to the 2D dataset also the 3D equivalent is designed for academic usage. The controlled data acquisition configuration restricts its applicability to real

world samples. Instead it allows to properly analyze effects due to the four environment / lightning conditions and object detection characteristics of the five categories. For the latter it is important to mention that the setup removes all possible context information and detection results can be solely attributed to the retrieval of object features. The main difference between the categories in the 3D domain is the number of points available per object. While a *table* object can be easily described by half of all points composing an acquired point cloud (e.g. see Figure 3.8-5), as less as three points can depict a *cup*. (The cup in Figure 3.8-2 is the largest used instance.) This makes the *cup* category especially important to analyze an algorithm's capacity to locate small and hardly captured objects. The main observed difference between the four conditions is that most tracking loss events happened in the *outdoor-sun* condition. A possible explanation for this effect is the difficult lightning in this condition. Reflections occur regularly because of the direct sun light. The number of mapping loss events also seems to correlate with the observed category. Most difficult is the *cup* category. This can most likely be explained by the challenging geometry. As stated above the point cloud often only contains the object and the platform below. With such a small object as a *cup* the point cloud mainly describes a flat surface. Having hardly a third dimension makes it difficult to anchor the point cloud within the real world and loosing or changing only a single point may therefore result into a mapping loss event.

3.3.3 2D - 3D Dataset properties

After comparing the 2D and 3D part of AR-2/3 with standard datasets from their respective domain, this Subsection discusses the similarities and differences between the 2D and 3D part. Most oft them have already been defined when detailed the dataset in Subsection 3.1.1 and 3.1.2.

As defined the number of categories is equal for the 2D and 3D part. The number of images is approximately twice as large as the number of point clouds, therefore also twice as much 2D than 3D objects are annotated. As most current 2D standard datasets are comprised of a significantly larger number of images and annotated objects than 3D datasets, this difference reflects the current state-of-the-art. Still the specifications of the 3D part are rather high, while the specification of the 2D part are rather low, thereby defining an acceptable compromise. Both datasets have the controlled environment and thereby the equally distributed number of annotations per category and condition as well as the high similarity between consecutive acquisitions in common. Condition and category specific variations are strongly domain specific (see subsections above) and no direct correlation can be observed.

CHAPTER 4

Analysis

Based on the presented the AR-2/3 dataset the comparison between 2D and 3D object detection can be performed. This chapter presents the applied methodology. Subsequently, first detection algorithm requirements are described, then the algorithm selection and tested configurations are presented, and finally, the comparison strategy is specified.

4.1 Algorithm Requirements

The general goal is to compare 2D and 3D object detection leveraging AR data. This comparison should be realized by setting up one representative 2D and one representative 3D object detection algorithm. Where the term *representative* stands for state-of-the-art algorithm in both accuracy and speed. As detailed in Section 2.1 and 2.2 best results in both the 2D and 3D domain are currently achieved by deep neural networks. Therefore DNNs should be used. Good algorithms are defined by high detection accuracy and low inference time. For the comparison between DNNs results on standard dataset — such as MS COCO (see Subsection 2.1.2) for 2D algorithms and KITTI / SUN RGB-D (see Subsection 2.2.2) for 3D algorithms — should be used. In detail for 2D comparison the MS COCO AP_{2D} value should be used as reference, as it summarizes Average Precision results for IoU values between 0.5 and 0.95. For the comparison performed in this thesis multiple IoU values should be employed (see Subsection 4.3) so the selected algorithm should perform well with different IoU values. In the 3D domain no multi IoU accuracy is available, in fact even multiple standard datasets are used. Therefore the mean Average Precision mAP_{3D} IoU 0.25 calculated either from KITTI or SUN RGB-D test set should be used. Single category results should be neglected as no global comparison between all describe algorithms is possible. Due to the domain driven development in the 3D domain it is also important to account for the selected categories. An algorithm developed for indoor detection may achieves better results on commodity object — as used in this experiment — than an algorithm developed for autonomous driving.

Additionally, the target application – real-time object detection with HoloLens 2 – must be considered. Therefore the selected algorithm should perform object detection in real-time. But, this does not imply that the algorithm has to run directly on HoloLens 2. A similar client-server setup as used for data acquisition (see Subsection 3.1.3) is conceivable. This would allow access to more processing capabilities than available on the device and — maybe most importantly — this would enable the usage of GPUs. Hence, more complex and resource demanding algorithm are an option, if they are fast enough.

To allow quick prototyping an easy to use, open-source implementation of the selected DNN should be available. An easy to use implementation should be well documented, actively developed, and use recent dependencies. Therefore enabling fast setup and access to support, in case of bugs or issues. Only open-source implementation should be considered as they are free of charge and the source code is publicly accessible. This allows for adaptations or extensions if required. Python implementations are preferred due to the author's experience with this programming language.

4.2 Algorithm Selection and Training

According to the defined requirements a 2D and 3D algorithm are selected and trained. The basis to realize the following comparison is the AR-2/3 dataset. In other words to allow a fair comparison both 2D and 3D algorithm must only use the respective data.

2D Data

The basis for 2D algorithm selection is the accuracy and speed comparison of state-of-the are CNNs shown in Table 2.2 and 2.3. In this comparison YOLOv4 reaches the highest accuracy in all categories and is also the second fastest algorithm behind YOLOv2. But, the original YOLOv4 paper was only published during test time of this master thesis. Therefore no easy to use implementation was available at that time. Hence, YOLOv4 is not considered in this thesis.

Second best algorithms are RetinaNet and YOLOv3. RetinaNet achieves better accuracy than YOLOv3, but is also significantly slower. Due to the real-time constraint RetinaNet is not suitable. In contrast, even YOLOv3's biggest version YOLOv3-608 reaches an acceptable inference time of 51ms. Additionally, the smaller — and faster — versions YOLOv3-416 and YOLOv3-320 reach better interference times of 29ms and 22ms respectively [Redmon and Farhadi, 2018]. For even faster processing YOLOv3-tiny [Huang et al., 2018] can be used. It is a lighter version of the full model containing less layers, only 23 instead of 106. Therefore inference is faster, only 1 GB instead of 4 GB of GPU RAM are required for training and it can be deployed to portable devices which do not have access to a GPU. The drawback is that it may miss some small objects [Yang et al., 2019]. At test time already a number of YOLOv3 open-source implementations were available.

Hence, YOLOv3 is selected as representative 2D object detection algorithm, achieving the best accuracy while also fulfilling the real-time constraint and the availability of an easy to use, open-source implementation. In detail both YOLOv3 and YOLOv3-tiny should be tested to additionally allow the comparison between an accuracy-focused and speed-focused implementation of a state-of-the-art 2D real-time object detection algorithm.

After evaluating a number of YOLOv3 implementations `yolov3-tf2`[1] is selected. It is a Python 3 - Tensorflow 2 implementation of both YOLOv3 and YOLOv3-tiny. The package is selected due to the author's experience with Python and Tensorflow which allowed the author to easily improve and extend the package e.g. by including data augmentation and parallelism of the training. Additionally, recent versions of package dependencies are used and the package itself is actively developed, allowing for support and bug fixes by the community.

`yolov3-tf2` is set up at Vienna Scientific Cluster 3. Up to 20 NVIDIA GeForce GTX 1080 based graphics card with 8GB of RAM each are used. The following configuration is used: Adam optimizer with $\beta = 0.9$, weight decay of 0.0005 and learning rate decay on plateau. Data augmentation is used for all training runs, implemented using `tf-image`[2]. In detail the following augmentation strategies were randomly applied: cropping, aspect ration distortion, quality reduction, erasing of image parts, rotation up to 45°, horizontal flipping, and color distortion (changing brightness, contrast, saturation, hue).

Leveraging this base configuration and AR-2/3, a number of hyperparameters are tested. In detail in this thesis different values for learning rate, batch size and input size are considered for both YOLOv3 and YOLOv3-tiny. Most studies training YOLOv3 and / or YOLOv3-tiny find that the best learning rates are 0.001 and 0.0001, see e.g. [Benjdira et al., 2019], [Ammar et al., 2019], and [Valiati and Menotti, 2019]. Therefore also this thesis focuses onto these two learning rates, but also tests 0.01 and 0.0001. For the batch size selection hardware constraints due to the available GPU RAM must be considered. Even though the original paper uses a batch size of 64 [Redmon and Farhadi, 2018], a number of other realizations also achieve good results with smaller values such as 24 [Thipsanthia et al., 2019], 4 [Rajendran et al., 2019] or even 2 [Valiati and Menotti, 2019]. Hence, due to the available GPU RAM of 8 GB, YOLOv3 focuses on training with the maximal possible batch size of 8 on a single GPU. To realize the large batch size of 64 training is parallelized onto eight GPUs. YOLOv3-tiny can — because of its lower requirements — easily be trained with a batch size of 64 even on a single GPU. Next to the batch size also the input size can affect the required GPU RAM. Most realizations use an input size of 416 as it is often a good trade-off between high accuracy and low inference times, see e.g. [Ammar et al., 2019]. That is why, this thesis focuses on an input size of 416, but also employs values of 320 and 608 to evaluate the effect. In addition to the discussed hyperparameteres, the usage of YOLOv3 and YOLOv3-tiny can be seen as another hyperparameter, as it varies the number of layers.

[1] https://github.com/zzh8829/yolov3-tf2, accessed 16-April-2021
[2] https://github.com/Ximilar-com/tf-image, accessed 14-May-2021

3D Data

For the selection of a representative 3D object detection algorithm Table 2.5 and 2.6 are used. Vote3Deep reaches the highest accuracy, but requires more then a second for inference. Therefore it does not fulfill the real-time constraint and is neglected. The second best algorithms are SECOND and VoteNet. The former is significantly faster while the latter achieve a slightly better accuracy. Another important difference between them is that SECOND is developed focusing on the autonomous driving application — evaluated on KITTI — while VoteNet development is mainly looking at commodity objects — evaluated on SUN RGB-D and ScanNet. Similar to VoteNet this thesis also focuses on commodity objects. Therefore, it is expected that VoteNet is able to better describe the dataset used in this thesis and achieves a higher accuracy. The original VoteNet implementation — `votenet`[3] — already covers all implementation requirements. Hence, VoteNet is used as 3D object detection algorithm.

The `votenet` package is a Python 3 - Pytorch 1.1 implementation. It has to be extended with an additional dataset class describing the AR-2/3 dataset (see `HoloexpDetectionVotesDataset`[4] and `HoloexpDatasetConfig`[5]). VoteNet expects point clouds and bounding boxes to be in a specific structure. The point cloud must be in an upright coordinate system, z-axis pointing upwards, y-axis pointing forward, and x-axis pointing to the right. Therefore HoloLens 2 point clouds — in forward coordinate system — are transformed. Also the data structure of VoteNet bounding boxes is defined clearly. They are described by the center coordinates (X, Y, Z), size (length, width, height) and heading angle around z-axis from positive x-axis to negative y-axis. Deriving the center coordinates is trivial. For the derivation of bounding box size and heading angle it must be considered that length and width can be defined arbitrarily, as long as the matching heading angle is selected. Figure 4.1 shows a 3D bounding box from top view. If v_1 is used as length, α must be used as heading angle, if v_2 is used as length β must be used as heading angle (see `VotenetLabelConverter`[6]). Based on the point cloud and bounding box votes are calculated.

Similar to `yolov3-tf2` also `votenet` is set up at Vienna Scientific Cluster 3 leveraging its NVIDIA GeForce GTX 1080 based graphics cards. The `votenet` implementation uses Adam optimizer with $\beta = 0.9$ and batch norm decay rate of 0.5 with a step size of 20. The network is trained for 180 epochs with learning rate decay at epoch 80, 120 and 160 with a decay rate of 0.1 each time. AR-2/3 is used as input dataset. Data augmentation is applied in an analogous manner as in the original paper [Qi et al., 2019]. This means, point clouds are randomly sub-sampled, randomly flipped in both horizontal directions,

[3] https://github.com/facebookresearch/votenet, accessed 17-May-2021

[4] https://git.geo.tuwien.ac.at/sherrmann/votenet/-/blob/master/holoexp/holoexp_detection_dataset.py, accessed 17-May-2021

[5] https://owly.duckdns.org/gitea/sophie/votenet/src/branch/master/holoexp/model_util_holoexp.py, accessed 17-May-2021

[6] https://git.geo.tuwien.ac.at/markus-kattenbeck/holo2_server/-/blob/master/app/resources/mesh/label_fmt_converter.py#L16, accessed 17-May-2021

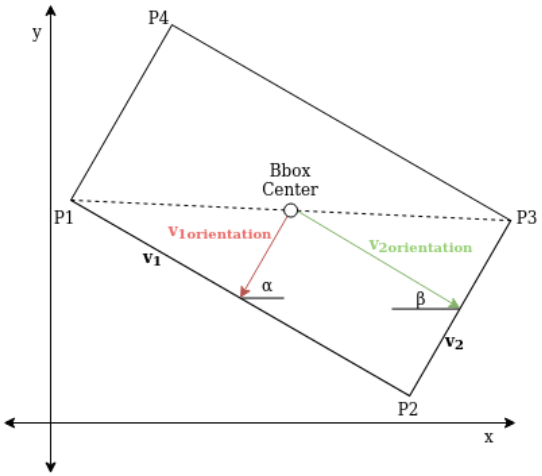


Figure 4.1: Definition of VoteNet bounding box size (v_1 / v_2) and heading angle (α / β) depend on each other. length = v_1, width = v_2 require heading angle = α while length = v_2, width = v_1 require heading angle = β.

points are randomly rotated around the z-axis (between -5° and 5°) and randomly scaled (between 0.9 and 1.1).

To optimize the model a number of hyperparameters combinations are tested. In detail different values for learning rate, batch size and number of input points are tested. In line with existing implementation learning rate values of 0.01 or 0.001 (see e.g. [Xie et al., 2020], [Qi et al., 2019]) as well as slightly bigger and smaller values (0.01 and 0.00001) are considered. A standard batch size of 8 — as used by all existing implementations (see e.g. [Xie et al., 2020], [Chen et al., 2021]) — is selected. Due to the limited GPU RAM of 8 GB no bigger batch size values can be tested, but also smaller values of 4 and 2 are taken into account. To describe the effect of point cloud size — alike the input size for YOLOv3 — the number of input points is varied. A standard value of 20,000 [Qi et al., 2019] as well as half (10,000) and double (40,000) the number is checked. To modify the network size, not only VoteNet, but also its simplified version BoxNet is used. In comparison to VoteNet BoxNet skips the voting step and infers categories and bounding boxes directly from the seed points [Qi et al., 2019]. Even though VoteNet achieves better results than BoxNet in the original paper, it seems that in special cases BoxNet performs better[7].

4.3 Comparison Strategy

Before comparing object detection accuracy results, the best 2D respectively 3D run, or the best set of hyperparameters, has to be identified. This decision is made based

[7] https://github.com/facebookresearch/votenet/issues/33, accessed 17-May-2021

on the achieved accuracy on the validation set of each model. In the 2D case again MS COCO's AP_{2D} is used, in the 3D case mAP_{3D} IoU 0.25 is leveraged. As `yolov3-tf2` only returns YOLOLoss but no accuracy metrics the package `review_object_detection_metrics` [8] [Padilla et al., 2021] is employed to calculate both PASCAL VOC AP_{2D} per category as well as MS COCO metrics AP_{2D}, AP_{2D-25}, AP_{2D-50}, and AP_{2D-75}. Minor improvements in the package are required to make it work with AR-2/3 [9]. `votenet` automatically calculates accuracy metrics, therefore no separate calculation is required.

The best 2D and 3D accuracy are then compared with results on standard datasets within the respective domain. In other words 2D results are set in relation to YOLOv3 results on MS COCO and 3D results are evaluated in contrast to VoteNet results on SUN RGB-D. This can give some insights into training success but may also describe the difficulty of the dataset. As the dataset shows a quite simple and constant scenario probably a higher accuracy can be achieved than on MS COCO and SUN RGB-D.

Finally, 2D and 3D object detection algorithms are evaluated. The comparison strategy focuses on two parts: accuracy and speed evaluation. Speed comparison can be easily done across the 2D and 3D domain. The mean inference time of all validation samples are calculated and compared. Such an across domain comparison is more demanding for accuracy values. As no previous research exists in across domain, 2D - 3D object detection accuracy comparison this thesis focuses on existing measures defined per domain. Most measures are defined equivalently across the domains, extending them with an additional dimension when moving from 2D to 3D. This implies, that for all comparison results it must be considered that 3D bounding boxes may be more difficult to derive and provide information about an additional dimension in respect to 2D bounding boxes.

In particular, again MS COCO metrics AP_{25}, AP_{50}, and AP_{75} are used as a basis. The central difference between the 2D and 3D implementation is that 2D accuracy is based on 2D IoU and 3D accuracy on 3D IoU. Because of the additional degrees of freedom high 3D accuracy is therefore more difficult to achieve than 2D accuracy. This strategy is selected even though also for 3D bounding boxes 2D accuracy metrics are available. But, the goal of thesis is to compare 2D and 3D object detection results and the first operates in the 2D domain and the latter in the 3D domain. It is argued that also the accuracy metric should therefore operated in the respective domain.

To evaluate effects of object properties and intra-category variabilities, PASCAL VOC AP_{2D} and AP_{3D} per category are used, again applying different IoU thresholds from 0.25 to 0.75. The per category results can give insights into effects due to object size, geometric complexity, and surface scale. Next to object centric effects also accuracy results per environment and lightning condition are presented and discussed.

[8] https://github.com/rafaelpadilla/review_object_detection_metrics, accessed 17-May-2021
[9] https://git.geo.tuwien.ac.at/sherrmann/review_object_detection_metrics, accessed 17-May-2021

CHAPTER 5

Results

The following chapter focuses on the results of the object detection experiment. As detailed in the previous chapter the 2D algorithm YOLOv3 and the 3D algorithm VoteNet are setup with 24 sets of hyperparameters each. For those setups validation accuracies are given. In particular, the overall accuracy as well as accuracy per category and condition are shown. For the best setup per domain also test accuracy and inference time are presented. First result for the 2D then for the 3D domain are given.

5.1 2D Data

To select the best set of hyperparameters for YOLOv3 accuracy metrics are calculated on validation data for each run. The results are presented in Figure 5.1, better values are visualized in green, worse in red. The best AP can be achieved with YOLOv3-tiny, a learning rate of 0.0001, a batch size of 64, and an input size of 416. Accuracy decreases with increasing IoU, though AP_{25} and AP_{50} are often quite similar, whereas AP_{75} is considerably below those. As AP is calculated from IoUs between 0.50 and 0.95 it focuses on rather high IoUs and reaches thereby lower results. Looking at the accuracy metrics per category most obvious is the high accuracy of the category *table* compared to all other categories. For IoU 0.25 and 0.5 the best six sets of hyperparameters achieve an average precision of over 0.9 for this category. Second best categories are most often *cup* and *pottedplant*. The most difficult category is *chair*. This is especially true for a large IoU of 0.75, where even the best set of hyperparameters reaches only an accuracy value of 0.1131. Evaluating the best sets of hyperparameters one can observed that smaller learning rates, larger batch sizes, and the medium input size achieve the best results.

As YOLOv3 is also trained with the smaller *labeling dataset* for the labeling procedure (discussed in Subsection 3.2.2) the validation accuracy are compared for initial evaluation. The best results for the labeling-dataset are summarized in Table 3.3 and 3.4. A first comparison shows that the achieved accuracy of the full dataset is considerably below

	Setting				mAP				IoU 0.25					IoU 0.5					IoU 0.75				
run	config	learning rate	batch size	input_size	AP	AP25	AP50	AP75	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table
41	tiny	0.0001	64	416	0.3596	0.6182	0.6147	0.3829	0.2194	0.4636	0.7950	0.6442	0.9691	0.2194	0.4587	0.7944	0.6432	0.9657	0.1131	0.1561	0.5785	0.4528	0.6295
40	tiny	0.001	64	416	0.3460	0.5932	0.5882	0.3699	0.0402	0.7525	0.3554	0.8745	0.9436	0.0402	0.7486	0.3554	0.8740	0.9379	0.0207	0.4428	0.2486	0.6067	0.5476
19	full	1E-05	8	416	0.3308	0.5156	0.5128	0.3845	0.0055	0.6796	0.0580	0.8809	0.9543	0.0055	0.6783	0.0580	0.8794	0.9502	0.0055	0.3674	0.0321	0.7024	0.8413
17	full	0.0001	8	416	0.2923	0.4606	0.4599	0.3311	0.1434	0.7468	0.1411	0.3393	0.9322	0.1434	0.7468	0.1411	0.3389	0.9302	0.1096	0.5030	0.1014	0.2441	0.7213
21	tiny	0.0001	8	416	0.2906	0.5048	0.5033	0.3049	0.0056	0.6132	0.3189	0.6735	0.9129	0.0056	0.6132	0.3189	0.6703	0.9101	0.0023	0.3491	0.2102	0.5009	0.4817
23	tiny	1E-05	8	416	0.2858	0.4982	0.4956	0.2987	0.0387	0.6738	0.4671	0.3800	0.9314	0.0387	0.6709	0.4671	0.3793	0.9266	0.0180	0.3379	0.3308	0.2546	0.5753
31	full	0.0001	4	608	0.2763	0.4456	0.4458	0.3176	0.0469	0.5311	0.2450	0.5670	0.8379	0.0469	0.5306	0.2446	0.5662	0.8351	0.0309	0.3280	0.1314	0.4507	0.6624
43	full	0.0001	64	416	0.2626	0.0000	0.5353	0.3583	0.1530	0.5643	0.4094	0.5350	0.7694	0.1321	0.5296	0.3576	0.5089	0.7509	0.0684	0.3002	0.2090	0.2812	0.5034
42	full	0.001	64	416	0.2215	0.5223	0.3851	0.2269	0.0182	0.6539	0.3813	0.6239	0.9344	0.0199	0.4792	0.2816	0.4495	0.6915	0.0142	0.2778	0.2430	0.3365	0.6826
33	tiny	0.0001	4	416	0.1974	0.3520	0.3515	0.1917	0.0053	0.5511	0.2557	0.0950	0.8528	0.0053	0.5511	0.2552	0.0950	0.8479	0.0016	0.3330	0.1946	0.0703	0.3758
24	full	0.001	4	416	0.1942	0.3171	0.3176	0.2150	0.0000	0.4743	0.2299	0.3318	0.5498	0.0000	0.4743	0.2291	0.3306	0.5465	0.0000	0.3104	0.1505	0.2674	0.3592
25	full	0.0001	4	416	0.1873	0.3085	0.3096	0.2054	0.0000	0.4779	0.2424	0.3307	0.4914	0.0000	0.4779	0.2419	0.3296	0.4887	0.0000	0.3010	0.1539	0.2626	0.3302
20	tiny	0.001	8	416	0.1702	0.2860	0.2863	0.1838	0.0012	0.0329	0.1232	0.3662	0.9064	0.0012	0.0323	0.1232	0.3650	0.9009	0.0005	0.0180	0.0966	0.2616	0.5453
29	full	0.0001	8	320	0.1098	0.4497	0.4007	0.0146	0.0725	0.8862	0.3303	0.1724	0.7871	0.0718	0.7602	0.2848	0.1509	0.7502	0.0000	0.0020	0.0035	0.0069	0.0581
37	tiny	0.0001	8	320	0.1064	0.3147	0.2995	0.0399	0.3421	0.2393	0.3424	0.1471	0.5028	0.3344	0.2104	0.3344	0.1345	0.4794	0.0976	0.0205	0.0138	0.0090	0.0540
38	tiny	0.001	8	608	0.1013	0.5585	0.4009	0.0106	0.4728	0.7891	0.4657	0.3291	0.7358	0.3860	0.7254	0.1960	0.2141	0.5112	0.0189	0.0114	0.0002	0.0045	0.0160
22	tiny	0.01	8	416	0.0854	0.1445	0.1467	0.0907	0.0000	0.0000	0.0243	0.1229	0.5754	0.0000	0.0000	0.0243	0.1224	0.5713	0.0000	0.0000	0.0206	0.0902	0.3262
39	tiny	0.0001	8	608	0.0655	0.5737	0.2980	0.0081	0.2414	0.7166	0.6414	0.4500	0.8190	0.2030	0.4798	0.1575	0.1793	0.5048	0.0139	0.0028	0.0001	0.0019	0.0115
16	full	0.001	8	416	0.0577	0.0982	0.1029	0.0549	0.0000	0.2125	0.2029	0.0011	0.0746	0.0000	0.2117	0.2029	0.0011	0.0746	0.0000	0.0717	0.1464	0.0011	0.0452
36	tiny	0.001	8	320	0.0542	0.1974	0.1789	0.0125	0.0365	0.0314	0.0746	0.1693	0.6753	0.0347	0.0295	0.0746	0.1187	0.6237	0.0028	0.0011	0.0025	0.0011	0.0400
28	full	0.001	8	320	0.0462	0.5307	0.1581	0.0073	0.0000	0.0000	0.0557	0.2821	0.5307	0.0000	0.0000	0.0553	0.2502	0.4807	0.0000	0.0000	0.0031	0.0067	0.0243
32	tiny	0.001	4	416	0.0431	0.0725	0.0751	0.0425	0.0002	0.0361	0.0329	0.0039	0.2893	0.0002	0.0361	0.0329	0.0039	0.2879	0.0001	0.0146	0.0284	0.0027	0.1554
18	full	0.01	8	416	0.0263	0.0375	0.0427	0.0303	0.0000	0.0011	0.0262	0.0687	0.0917	0.0000	0.0011	0.0262	0.0687	0.0917	0.0000	0.0007	0.0173	0.0392	0.0683
30	full	0.001	4	608	0.0162	0.0837	0.0505	0.0032	0.0000	0.2075	0.0354	0.0155	0.1604	0.0000	0.2032	0.0091	0.0058	0.0301	0.0000	0.0101	0.0000	0.0000	0.0000

Figure 5.1: Validation accuracy metric YOLOv3 for defined 24 sets of hyperparameters sorted by AP in descending order. Color scale: red ≤ 0.4, yellow $= 0.7$, green ≥ 1. Most runs achieve only a very poor accuracy.

the one of the labeling dataset, even though the full dataset includes a higher number of samples (more then 112,000 vs 5,000 train samples). To better describe possible causes of this difference three variations between the training with the labeling dataset and training with the full dataset are considered. First, bounding boxes in the labeling dataset are defined manually and not through an automatic labeling process based on YOLOv3 itself. Therefore bounding boxes are likely more precise in the labeling dataset. But, as also mean average precision with an IoU of 0.25 is significantly lower of the full dataset this difference can be neglected. Second, for the training with the labeling dataset transfer learning is used, thereby accelerating training as previously learned knowledge can be used. This could have an effect on the result, but to achieve 2D - 3D comparable results transfer learning is not an option for this experiment. Third, no data augmentation is used for the labeling dataset. Due to the well defined data acquisition setup the dataset is quite homogeneous. Therefore, the application of data augmentation could affect the final accuracy.

To evaluate the effect of data augmentation on the accuracy, the best five sets of hyperparameter are trained with two additional levels of data augmentation *medium* and *no*. In this thesis medium data augmentation includes random erasing, random rotation by up to 15°, and random color distortion (changing brightness, contrast). For the no data augmentation level only the original images are used. The originally applied level of data augmentation is called *full* data augmentation. After training the additional models, again the validation accuracy per category and per condition are calculates. For the model with the highest accuracy per data augmentation level then also the test accuracy per category and per condition are calculated. This complete YOLOv3 workflow is visualized in Figure 5.2.

The validation accuracy results per category are shown in Figure 5.3. The same color scale as in Figure 5.1 is used, visualizing better values in green and worse values in red. Generally, one can observe an accuracy increase from full, to medium and no data augmentation. Considering all five sets of hyperparameters applying no augmentation seems to achieve the best results. But, the overall best model is trained in run 53 with medium data augmentation. It is slightly better than the best no data augmentation in all four mean Average Precision values (AP, AP_{25}, AP_{50}, AP_{75}) presented. Interestingly the best set of hyperparameter is equal for all three levels of data augmentation and it is the tiny version of YOLOv3. In an analogues manner to full data augmentation AP_{25} and AP_{50} achieve similar results also with the additional two levels of data augmentation. Different is the accuracy distribution over the categories especially for the best runs. For an IoU of 0.25 and 0.5 nearly all categories achieve Average Precision of over 0.9, only the category *chair* - medium data augmentation reaches with 0.8453 a value below (run 53). With the largest IoU of 0.75 there are also larger differences between the accuracy values per category. Most obviously is again the lower accuracy of category *chair*. This is similar to the behavior observed for full augmentation. In summary, category *chair* seems to be most difficult to detect, especially with a large IoU.

Next to the possibility to evaluate accuracy per category, with this dataset it is also

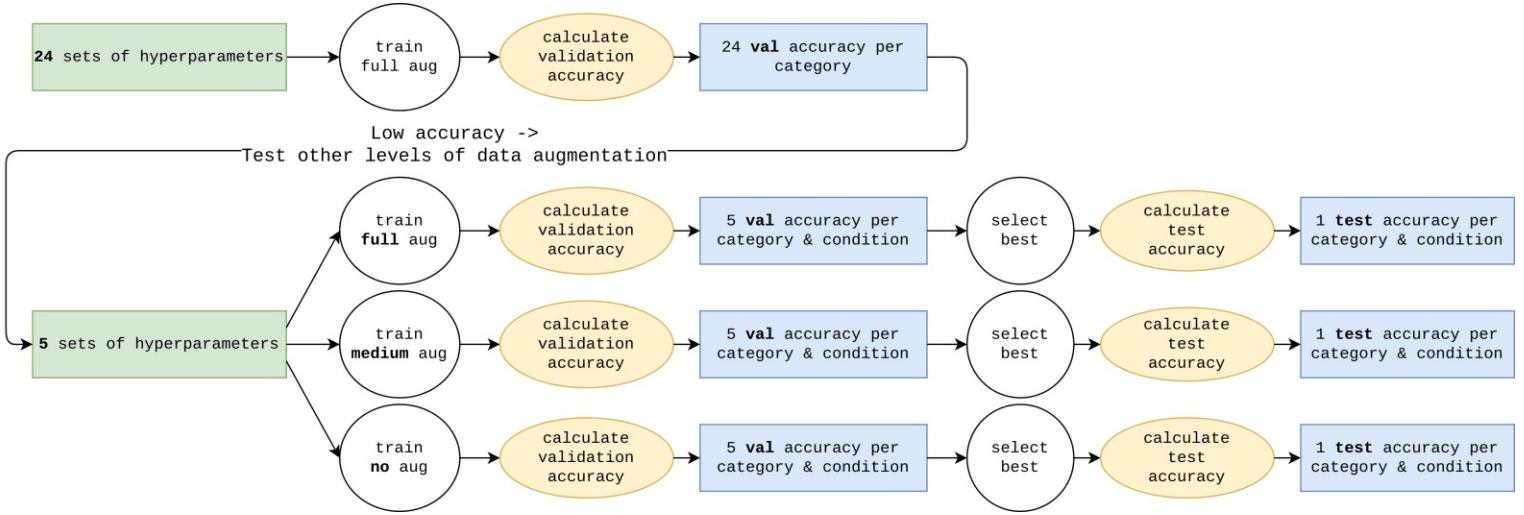


Figure 5.2: YOLOv3 processing and evaluation workflow. Due to the poor results based on the initial set of augmentation tasks in total three levels of data augmentation are evaluated.

		Setting				mAP				IoU 0.25					IoU 0.5					IoU 0.75				
	run	config	learning r	batch size	input_size	AP	AP25	AP50	AP75	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table
full aug	41	tiny	0.0001	64	416	0.3596	0.6182	0.6147	0.3829	0.2194	0.4636	0.7950	0.6442	0.9691	0.2194	0.4587	0.7944	0.6432	0.9657	0.1131	0.1561	0.5785	0.4528	0.6295
	40	tiny	0.001	64	416	0.3460	0.5932	0.5882	0.3699	0.0402	0.7525	0.3554	0.8745	0.9436	0.0402	0.7486	0.3554	0.8740	0.9379	0.0207	0.4428	0.2486	0.6067	0.5476
	19	full	1E-05	8	416	0.3308	0.5156	0.5128	0.3845	0.0055	0.6796	0.0580	0.8809	0.9543	0.0055	0.6783	0.0580	0.8794	0.9502	0.0055	0.3674	0.0321	0.7024	0.8413
	17	full	0.0001	8	416	0.2923	0.4606	0.4599	0.3311	0.1434	0.7468	0.1411	0.3393	0.9322	0.1434	0.7468	0.1411	0.3389	0.9302	0.1096	0.5030	0.1014	0.2441	0.7213
	21	tiny	0.0001	8	416	0.2906	0.5048	0.5033	0.3049	0.0056	0.6132	0.3189	0.6735	0.9129	0.0056	0.6132	0.3189	0.6703	0.9101	0.0023	0.3491	0.2102	0.5009	0.4817
Medium Aug	53	tiny	0.0001	64	416	0.5961	0.9445	0.9389	0.6818	0.8453	0.9596	0.9736	0.9550	0.9888	0.8453	0.9585	0.9736	0.9545	0.9862	0.5229	0.6754	0.8216	0.7873	0.6308
	50	full	1E-05	8	416	0.5256	0.7740	0.7712	0.6249	0.4984	0.9631	0.6957	0.8379	0.8749	0.4984	0.9631	0.6953	0.8372	0.8722	0.4307	0.7917	0.5681	0.7176	0.6631
	54	tiny	0.001	64	416	0.4870	0.7966	0.7932	0.5439	0.4212	0.8598	0.7804	0.9561	0.9658	0.4212	0.8557	0.7797	0.9554	0.9615	0.1940	0.6605	0.5818	0.7762	0.5409
	51	tiny	0.0001	8	416	0.4823	0.7887	0.7843	0.5301	0.7644	0.8957	0.7529	0.5739	0.9567	0.7644	0.8957	0.7517	0.5734	0.9525	0.4262	0.6181	0.5822	0.4555	0.5997
	52	full	0.0001	8	416	0.4460	0.6690	0.6656	0.5320	0.6663	0.6289	0.3618	0.8157	0.8721	0.6663	0.6284	0.3613	0.8147	0.8687	0.5734	0.4515	0.2646	0.6923	0.7023
no aug	11	tiny	0.0001	64	416	0.5954	0.9605	0.9545	0.6509	0.9739	0.9521	0.9600	0.9443	0.9719	0.9734	0.9513	0.9589	0.9443	0.9666	0.5516	0.6488	0.7255	0.7465	0.6362
	7	full	1E-05	8	416	0.5625	0.9314	0.9256	0.6063	0.9029	0.9767	0.8889	0.9414	0.9469	0.9029	0.9743	0.8877	0.9398	0.9447	0.5427	0.6845	0.5597	0.6299	0.6483
	15	tiny	0.0001	8	416	0.5293	0.8591	0.8549	0.5749	0.9005	0.9038	0.8457	0.7129	0.9326	0.8998	0.9028	0.8446	0.7129	0.9272	0.5094	0.6567	0.6406	0.5521	0.5367
	6	full	0.0001	8	416	0.4874	0.7804	0.7781	0.5357	0.8159	0.5036	0.6407	0.9694	0.9726	0.8153	0.5032	0.6395	0.9685	0.9702	0.5650	0.3265	0.3665	0.7284	0.7217
	9	tiny	0.001	64	416	0.4066	0.7391	0.7335	0.3874	0.5532	0.5236	0.6889	0.9676	0.9622	0.5532	0.5203	0.6884	0.9653	0.9539	0.2294	0.3179	0.3756	0.6181	0.4291

Figure 5.3: Validation accuracy of best five sets of hyperparameters for three levels of augmentation (full, medium, no). The results are sorted per augmentation level by *AP*. Run 53 (medium aug.) achieve the best accuracy. Generally, medium and no data augmentation perform significantly better than full data augmentation.

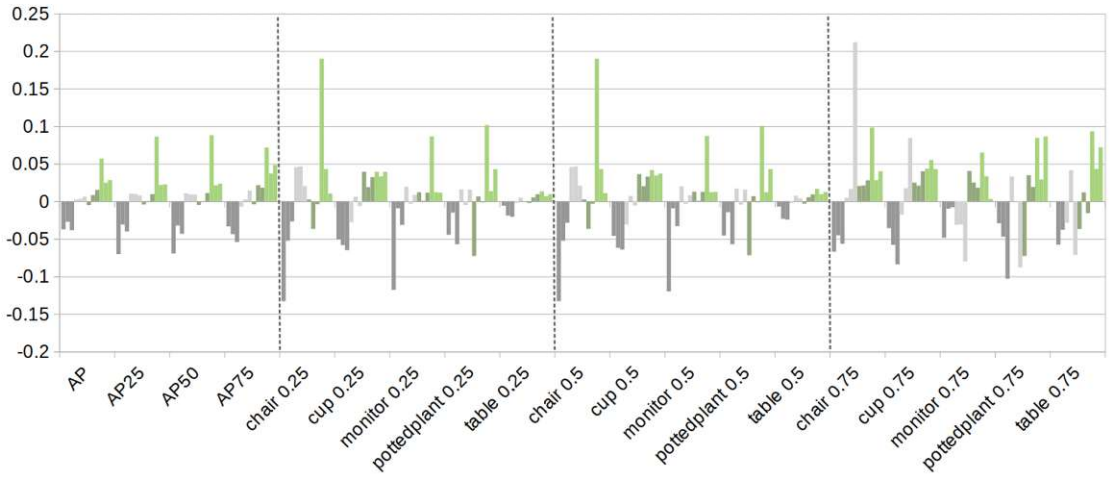


Figure 5.4: Differences between the four conditions and overall accuracy results on the validation dataset. Negative values denote worse, positive values denote better condition results. (dark-grey: indoor-night, light-grey: indoor-sun, dark-green: outdoor-night, light-green: outdoor-sun) Results are given for all three level of augmentation, order from left to right: full - medium - no augmentation per accuracy metric. *Indoor-night* generally seems to be the most demanding and *outdoor-sun* the simplest condition.

possible to examine the effect of four environmental and lightning conditions (*indoor-night*, *indoor-sun*, *outdoor-night*, and *outdoor-sun*, see Subsection 3.1.2). To better visualize changes in accuracy between the different conditions Figure 5.4 shows the difference between the condition specific accuracy results and the overall validation accuracy results for the best run per augmentation level (order from left to right full - medium - no augmentation). Negative values, therefore, denote that the condition specific accuracy is lower / worse than the overall one, and positive values denote that the condition specific accuracy is higher / better than the overall one. Detailed results for the best five runs per augmentation level can be found in the Appendix 7 Figure 1, 2, and 3.

One apparent observation is that for all data augmentation levels — but most pronounced on data augmentation level full (left most bar per accuracy and condition) — *mAP* values are better for the condition *outdoor-sun* and worse for the condition *indoor-night*. The *indoor-sun* and *outdoor-night* condition are quite close to the overall results. Looking at the per category accuracy values one can observe more variations. For nearly all categories and IoUs *indoor-night* is the worst and *outdoor-sun* is the best condition for all augmentation levels. The results for the two other conditions vary more than for *mAP* results and may differ strongly between the augmentation levels (e.g. condition *indoor-sun*, category *table*, IoU 0.75).

For the best model per data augmentation level also the test accuracy is calculated. Test accuracy is extremely similar to validation accuracy for all different IoU values, categories and conditions (see Figure 5.6). Interestingly run 11 (no data augmentation) reaches

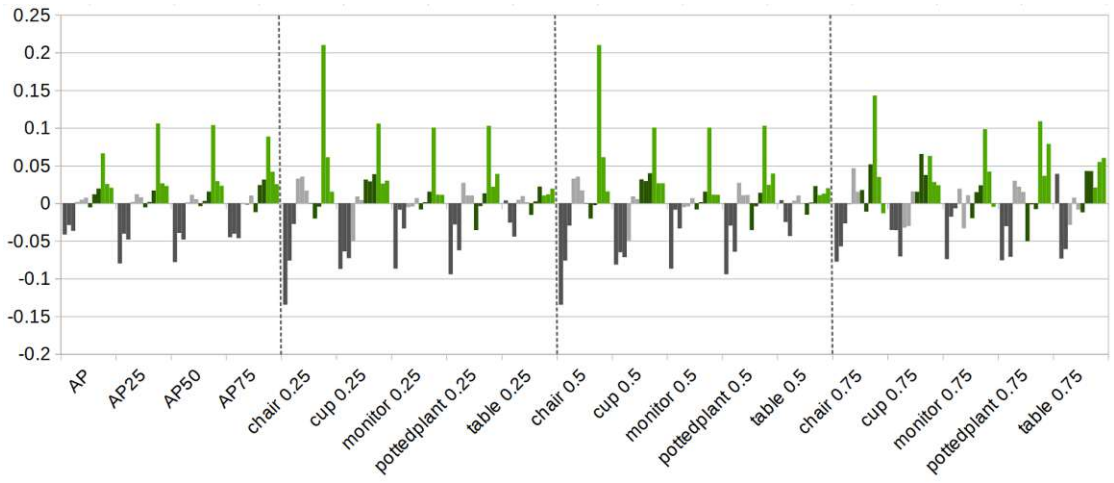


Figure 5.5: Differences between the four conditions and overall accuracy results on the test dataset. Negative values denote worse, positive values denote better condition results. (dark-grey: indoor-night, light-grey: indoor-sun, dark-green: outdoor-night, light-green: outdoor-sun) Results are given for all three level of augmentation, order from left to right full - medium - no augmentation per accuracy metric. Generally, results from the *indoor-night* condition are worst and results from the *outdoor-sun* condition are best.

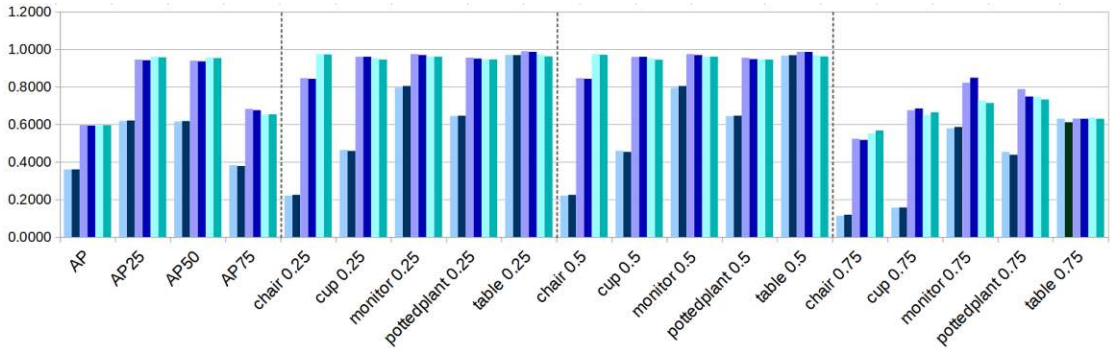


Figure 5.6: Overall accuracy of best run on validation and test data. For each accuracy validation (light) and test (dark) results are shown, full: blue, medium: purple, no: turquoise. Full augmentation performs worse, medium and full augmentation are considerable better and are quite similar.

slightly the highest test accuracy, run 53 (medium data augmentation) is only second best. Figure 5.5 shows the differences from the overall to the condition specific results for the test dataset. Those results for the test dataset are very similar to the validation dataset ones. On NVIDIA GeForce GTX 1080 based graphic cards and an input size of 416 an inference time of 0.0155 seconds could be reached for YOLOv3-tiny, YOLOv3 requires 0.0516 seconds per image.

5.2 3D Data

Also for the 3D data, validation accuracy is calculated for each set of tested hyperparameters. The results are shown in Figure 5.7 sorted from highest to lowest AP_{25}. Run 21 achieves with configuration votnet, learning rate 0.001, batch size 8 and 40,000 input points the highest AP_{25} and AP_{50}. Only in AP_{75} run 1 — configuration votenet, learning rate 0.001, batch size 8, number of point 20,000 — reaches better results. Similar to the 2D domain accuracy decreases with increasing IoU and while AP_{25} and AP_{50} are quite close, there is a larger difference to AP_{75}. For the first two, nearly always values above 0.9 are achieved. In comparison the best AP_{75} is 0.7191 (run 1). Looking at the per category results one can observe that the category *cup* achieves always the worst accuracy. With an IoU of 0.25 also the *cup* category can reach an accuracy of 0.9662, but with an IoU of 0.75 even the best model achieves only a value of 0.2001 (run 1). All other four categories achieve similar and good results for IoU 0.25 and 0.5. Difference between those categories can only be observed for results based on IoU 0.75. Even with this high IoU the detection of category *pottedplant* performs well. The categories *chair* and *monitor* are second best, and the category *table* performs a bit worse compared to those two.

Differences between the overall accuracy and the per condition results are very similar for the different runs, therefore only the results for the best run 21 are shown in Figure 5.8. The complete set of deviations can be found in the Appendix 7 Figure 5 - 8. Different than in the 2D domain no systematic effect due to the conditions can be observed, but rather only larger variations per category - condition combinations. For example, the overall most difficult cup category performs better in the two indoor conditions and worse in the outdoor condition, while the *table* category performs worst in the *indoor-night* condition and better in the other three. Interestingly there can also be variations between the results for different IoU values. For the *cup* category the variations are largest for an IoU of 0.5. In contrast, for the other four categories hardly any variations are observed for IoU 0.25 and 0.5, but large variations are apparent for an IoU of 0.75.

For the best model (run 21) accuracy is calculated on the test set. Overall accuracy results and deviations for the four conditions are summarized in Figure 5.10 respectively 5.9. Overall results are similar but mostly slightly worse than validation accuracy. Only some categories for some IoU values reach a bit higher accuracy values, e.g. category *chair* for IoU 0.5 and 0.75 performs slightly better on the test than on the validation set. Condition specific differences show a similar distribution on the test as on the validation set. The inference time is again tested on NVIDIA GeForce GTX 1080 based graphic cards. The best VoteNet model reaches a value of 0.1904 seconds.

	Settings				mAP			IoU 0.25					IoU 0.5					IoU 0.75				
run	config	learning rate	batch size	num_points	AP25	AP50	AP75	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table
21	votenet	0.001	8	40000	0.9927	0.9671	0.7073	1.0000	0.9662	1.0000	1.0000	0.9971	0.9974	0.8501	1.0000	0.9983	0.9899	0.8590	0.1455	0.8153	0.9569	0.7597
9	votenet	0.001	4	20000	0.9923	0.9610	0.6854	0.9996	0.9666	0.9995	0.9993	0.9964	0.9969	0.8218	0.9984	0.9984	0.9894	0.8159	0.1790	0.7846	0.9571	0.6903
13	votenet	0.001	2	20000	0.9892	0.9441	0.5804	0.9976	0.9602	0.9912	0.9993	0.9975	0.9960	0.7576	0.9796	0.9983	0.9889	0.7728	0.0656	0.7122	0.7909	0.5604
1	votenet	0.001	8	20000	0.9887	0.9594	0.7191	0.9841	0.9633	0.9999	1.0000	0.9961	0.9811	0.8328	0.9984	0.9974	0.9872	0.8539	0.2001	0.8218	0.9514	0.7684
17	votenet	0.001	8	10000	0.9878	0.9546	0.7067	0.9989	0.9431	0.9993	0.9993	0.9984	0.9969	0.7874	0.9986	0.9984	0.9919	0.8369	0.1474	0.8204	0.9601	0.7685
19	boxnet	0.001	8	10000	0.9841	0.9344	0.6321	0.9996	0.9235	0.9993	1.0000	0.9979	0.9973	0.6863	0.9985	0.9993	0.9904	0.7981	0.0133	0.7279	0.9465	0.6747
5	boxnet	0.001	8	20000	0.9833	0.9321	0.6340	1.0000	0.9205	0.9986	1.0000	0.9971	0.9991	0.6736	0.9978	0.9993	0.9908	0.8040	0.0184	0.7485	0.9289	0.6702
3	votenet	0.01	8	20000	0.9832	0.9309	0.5795	1.0000	0.9227	0.9992	0.9993	0.9949	0.9961	0.6942	0.9892	0.9983	0.9768	0.7837	0.0725	0.6671	0.9517	0.4225
23	boxnet	0.001	8	40000	0.9828	0.9357	0.6425	1.0000	0.9174	0.9986	1.0000	0.9979	0.9992	0.6961	0.9970	0.9993	0.9869	0.8287	0.0182	0.7646	0.9498	0.6512
22	votenet	0.0001	8	40000	0.9816	0.9189	0.5631	1.0000	0.9137	0.9966	1.0000	0.9978	0.9973	0.6233	0.9902	0.9982	0.9853	0.7248	0.0302	0.5438	0.9304	0.5863
18	votenet	0.0001	8	10000	0.9811	0.9229	0.5774	0.9991	0.9147	0.9974	0.9993	0.9952	0.9959	0.6495	0.9882	0.9975	0.9835	0.7351	0.0479	0.5717	0.9408	0.5918
20	boxnet	0.0001	8	10000	0.9810	0.8982	0.5212	1.0000	0.9084	0.9985	1.0000	0.9979	0.9982	0.5151	0.9953	0.9967	0.9859	0.7057	0.0005	0.5562	0.9178	0.4259
7	boxnet	0.0001	8	20000	0.9806	0.8949	0.5203	0.9999	0.9072	0.9986	1.0000	0.9971	0.9978	0.4966	0.9963	0.9975	0.9864	0.6897	0.0004	0.5833	0.9139	0.4142
12	boxnet	0.0001	4	20000	0.9803	0.9105	0.4858	1.0000	0.9144	0.9900	1.0000	0.9971	0.9981	0.5954	0.9885	0.9822	0.9883	0.6792	0.0051	0.6130	0.6984	0.4333
24	boxnet	0.0001	8	40000	0.9792	0.9052	0.5272	1.0000	0.9025	0.9971	0.9993	0.9971	0.9980	0.5562	0.9964	0.9981	0.9770	0.7079	0.0018	0.6111	0.9326	0.3825
6	boxnet	0.01	8	20000	0.9792	0.9110	0.5616	1.0000	0.9018	0.9971	1.0000	0.9971	0.9980	0.5907	0.9927	0.9981	0.9756	0.7583	0.0025	0.6797	0.9492	0.4186
11	boxnet	0.001	4	20000	0.9791	0.9278	0.5584	1.0000	0.8991	0.9979	1.0000	0.9986	1.0000	0.6874	0.9942	0.9688	0.9885	0.7637	0.0212	0.7349	0.6474	0.6249
14	votenet	0.0001	2	20000	0.9789	0.9340	0.5894	0.9996	0.9035	0.9956	0.9993	0.9964	0.9983	0.6934	0.9920	0.9976	0.9884	0.7779	0.0517	0.6474	0.9129	0.5572
16	boxnet	0.0001	2	20000	0.9751	0.8742	0.2762	0.9999	0.8890	0.9907	0.9993	0.9964	0.9941	0.6576	0.9907	0.7602	0.9683	0.5166	0.0059	0.4971	0.0702	0.2912
10	votenet	0.0001	4	20000	0.9749	0.9221	0.5807	1.0000	0.8862	0.9913	0.9993	0.9978	0.9947	0.6499	0.9818	0.9985	0.9858	0.7667	0.0468	0.6684	0.8753	0.5466
15	boxnet	0.001	2	20000	0.9745	0.8618	0.3189	1.0000	0.8967	0.9907	0.9886	0.9964	0.9981	0.6947	0.9892	0.6357	0.9911	0.5199	0.0098	0.5913	0.0824	0.3910
2	votenet	0.0001	8	20000	0.9744	0.9103	0.5885	0.9998	0.8769	0.9993	0.9998	0.9961	0.9986	0.5690	0.9993	0.9978	0.9868	0.7752	0.0367	0.6402	0.9413	0.5492
4	votenet	1E-05	8	20000	0.9636	0.8180	0.2726	1.0000	0.8501	0.9810	0.9986	0.9885	0.9966	0.3402	0.9234	0.9947	0.8350	0.4008	0.0002	0.1506	0.7305	0.0811
8	boxnet	1E-05	8	20000	0.9212	0.7610	0.1763	0.9999	0.6332	0.9871	0.9999	0.9861	0.9968	0.0744	0.9028	0.9947	0.8362	0.2164	0.0000	0.0954	0.5529	0.0169

Figure 5.7: Validation accuracy VoteNet for defined 24 sets of hyperparameters sorted by AP_{25} in descending order. Color scale: red ≤ 0.4, yellow $= 0.7$, green ≥ 1. For most runs accuracy is good. Category *cup* performs worse than all other categories.

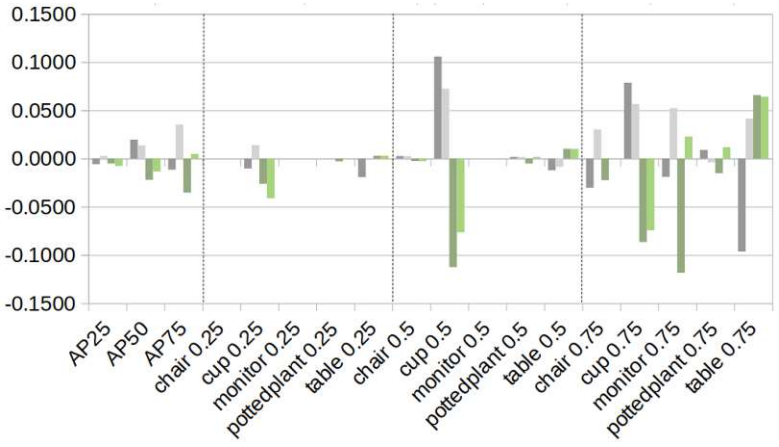


Figure 5.8: Differences between the four conditions and overall accuracy results on the validation dataset. Negative values denote worse, positive values denote better condition results. (dark-grey: indoor-night, light-grey: indoor-sun, dark-green: outdoor-night, light-green: outdoor-sun). Largest variation can be observed for category *cup*, for this category indoor conditions are considerably better than outdoor conditions.

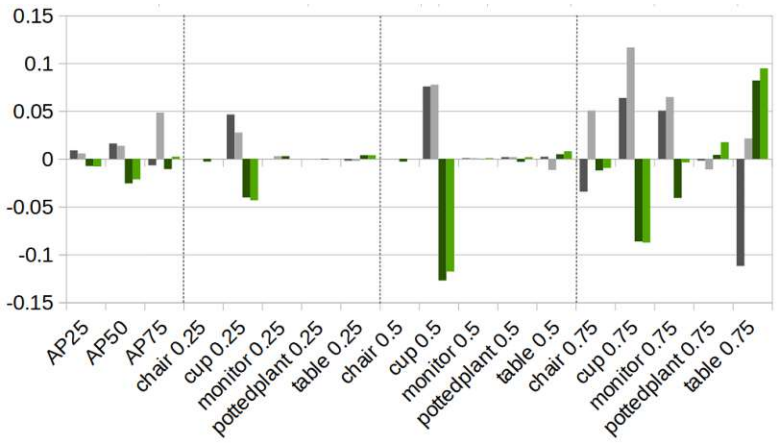


Figure 5.9: Differences between the four conditions and overall accuracy results on the test dataset. Negative values denote worse, positive values denote better condition results. (dark-grey: indoor-night, light-grey: indoor-sun, dark-green: outdoor-night, light-green: outdoor-sun)

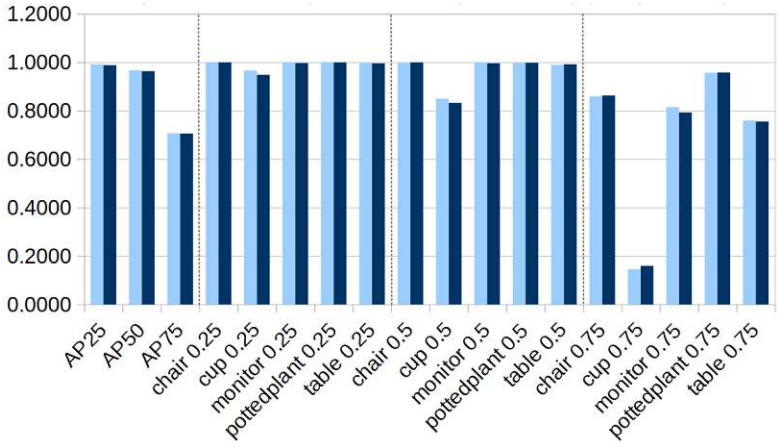


Figure 5.10: Overall accuracy of best run on validation and test data. Results are very similar between validation and test set.

CHAPTER 6

Discussion

After the presentation of the results of the object detection experiment the following chapter discusses those results. The first part focuses on a per domain evaluation and the second part, finally, takes a close look at the comparison between the 2D and 3D domain.

6.1 Object Detection Results per Domain

The following section evaluates the object detection results for the 2D and 3D domain separately. The focus is on the comparison to results of the algorithms on standard datasets and variations between categories as well as conditions.

6.1.1 2D Data

The results are first evaluated in comparison to results on MS COCO. Figure 6.1 shows test AP_{25}, AP_{50}, AP_{75}, and AP for the three applied data augmentation levels and the results for MS COCO (taken from [Redmon and Farhadi, 2018, p. 3]). For all data augmentation levels an IoU of 0.25 and 0.5 achieve similar accuracy results. As the smallest used IoU in the 2D domain is normally 0.5, models are probably designed to at least provide acceptable results with this minimum IoU. The drop in accuracy for higher IoU values is in accordance to the original paper, also identifying a drop in accuracy when increasing the IoU. Due to the simplicity of the new dataset AR-2/3, accuracy results are expected to be higher on this dataset than on MS COCO — as also achieved on the labeling dataset — but for full augmentation results are similar to the results achieved with MS COCO. Medium and no data augmentation are similar and higher by a value of approximately 0.3.

This effect of decreasing accuracy for full augmentation level is also visible from the more extensive validation results (see Figure 5.3) and the test results (see Figure 5.6). Such results are unexpected as data augmentation is normally used to improve generalization

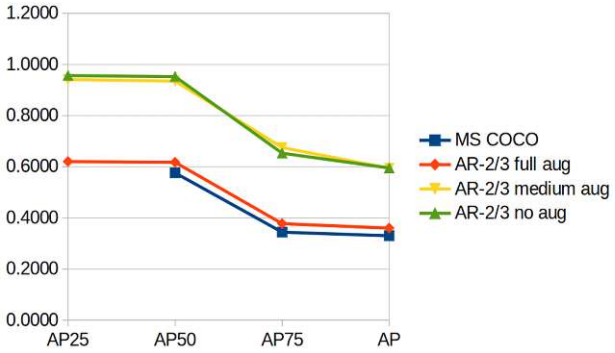


Figure 6.1: Average Precision of best YOLOv3 test results for different data augmentation levels on AR-2/3 in comparison with results on MS COCO. Augmentation level medium and no perform considerably better than augmentation level full and evaluation on MS COCO.

capabilities of a network. Strong empirical evidence has been provided for the positive effects of data augmentation on image classification in a variety of studies (e.g. [Perez and Wang, 2017] or [Mikołajczyk and Grochowski, 2018]). Data augmentation for object detection is especially for geometric distortions more challenging but nowadays a number of publications exist describing the advantages also for this task (e.g. [Shorten and Khoshgoftaar, 2019] or [Zhong et al., 2020]). Nevertheless researches apply different numbers of data augmentation techniques. For instance YOLO [Redmon et al., 2016] and its successor models YOLOv2 [Redmon and Farhadi, 2017] and YOLOv3 [Redmon and Farhadi, 2018] leverage a large number of data augmentation techniques such as random crops, color shifting, and translations. In contrast, RetinaNet [Lin et al., 2017b] applies solely horizontal flipping.

[Zoph et al., 2020] analyze whether it is possible to further improve RetinaNet's accuracy by just optimizing the applied augmentation tasks. They are able to achieve an accuracy increase of 2.3 on mAP. This and other studies (e.g. [Cubuk et al., 2018]) show that the best set of augmentation tasks often depends on the data itself. To automatically determine the best set of augmentation tasks for a given dataset before training, a number of algorithms have been developed (e.g. [Zoph et al., 2020], [Lim et al., 2019]). In this thesis' experiment no prior evaluation of the best set of data augmentation task is performed neither for full nor for medium data augmentation. Therefore, it is possible that — by chance — the medium augmentation level defines a better set of augmentation tasks. Though the difference in this experiment is by a factor of ten higher than in [Zoph et al., 2020] analysis, it could be a contributing factor.

It is argued that the main reason for the low accuracy of the full augmentation level is the very high number of augmentation tasks applied to each image. As described in 4.2 all used augmentation tasks are applied with a probability of 50% per image and epoch. Hence, with ten augmentation tasks — as used in full augmentation — it is very

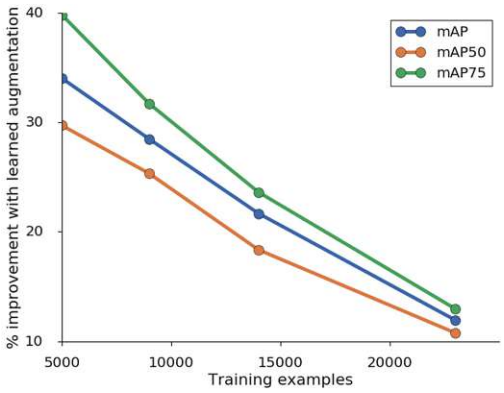


Figure 6.2: The importance of data augmentation decreases with increasing number of input samples. Additionally it is more important for higher IoU values. (Figure taken from [Zoph et al., 2020, p. 578]

unlikely that no augmentation task is applied to a single image. That is why, the model sees hardly any original / not augmented image during the training process. In most cases both geometric and color distortions are leveraged. This makes it hard for the model to generalize category properties to not augmented representations. Therefore, detecting them in the validation or test dataset is difficult for the model. In contrast, using medium data augmentation only four augmentation tasks are used, thereby some properties of the original images are most often preserved and the accuracy significantly increases. Also research focusing on data augmentation optimization often only apply a low number of augmentation tasks per image. E.g. [Zoph et al., 2020] and [Cubuk et al., 2020] apply two augmentation tasks with a probability of 50% each per image. Though it should be mentioned that — different than in this thesis — they pick those two from a larger pool of augmentation tasks.

Moreover, previous research shows that data augmentation is most important and efficient for originally small datasets and difficult detections [Zoph et al., 2020]. The reduced importance of data augmentation for an increasing number of training samples is visualized in Figure 6.2. At this point it is important to notice that in the Figure a maximum number of 20,000 training samples were used, but the experiment presented in this thesis uses more then 112,000 training samples. Considering the graph suggest that with such a large number of sample even a good set of augmentation tasks achieves only a small accuracy improvement. This explains the extreme similarity between medium and no data augmentation. Moreover, Figure 6.2 documents that the importance of data augmentation increases for more strict IoUs. This effect is also apparent for results of this dataset, AP_{75} — the highest considered IoU — benefits most from data augmentation and therefore medium performs better than no data augmentation (see Figure 6.1).

Another important characteristic of the dataset in this context is its simplicity because of the small number of categories, the large number of samples, and the static observation scenario. This generally explains the higher accuracy results achieved on this dataset

compared to MS COCO. It is also a starting point to further discuss the bad results of the full data augmentation level. Because of the size of the dataset and its homogeneity the necessity and usefulness of data augmentation is questionable. As validation and test datasets contain structurally very similar samples like the training dataset it could be argued that the model can learn the most for this population from the original samples.

An evaluation of the best hyperparameters shows that all augmentation levels perform best with the same set of hyperparameters. Interestingly the simpler tiny version performs better than full YOLOv3. This can again be explained by the simplicity of the dataset. The smaller network is sufficient to describe but also to abstract the — only — five categories with its iconic objects. Larger batch size (64) and small learning rate (0.0001 and 1e-5) perform best. One possible explanation is that there are many very similar images in that dataset so it is better to see a large number of images before updating network weights. Additionally, it is better to perform rather small updates especially if the batch size is smaller. With 416 the best input size is the medium option which is also preferred by most previous studies, see e.g. [Ammar et al., 2019]. The lower input size of 320 apparently does not provide enough information. The higher input size of 608 is limited by the original resolution of the images (896 × 504 pixels) and does not add important information. Moreover for the larger input size the training time increases significantly.

After discussing the best hyperparameters the accuracy results per category are evaluated. Category *table* could be learned best by all levels of augmentation. This is an expected behavior as it is the category with the significantly largest instances. Because of the data acquisition configuration object instances of this category also cover large parts of acquired *table*-images and large objects on images are easier to detect for YOLO than small ones [Redmon and Farhadi, 2018]. *Cup*, *monitor*, and *pottedplant* achieve similar results considering all five best sets of hyperparameters. Only the accuracy of category *chair* is quite low for most runs. This is unexpected as *chair* is a quite big category, which should ensure good detection results. An exception to the low *chair* accuracy is run 11 (no augmentation) where the accuracy of category *chair* is comparable to the other categories and mAPs. As shown in Figure 3.12a category *chair* is described by the least number of images. Moreover, it is the only category which looks different from each side — neglecting the handle of a cup as it is not sure whether the handle is considered for detecting cups and neglecting pottedplants which are not perfectly symmetric but the rough structure is consistent. Depending on the perspective a chair's backrest changes the overall geometry of a chair. Some chair instances are made of thin planks. Looking at such instances from a side view, the chair is despite its size only described by thin structures which may be difficult to distinguish from the background. Neutral colors such as brown or gray further support this theory. To confirm this assumptions more research is required analyzing object properties of this category (see Chapter 7).

Category level evaluation allows conclusions about the importance of some object properties and intra-category variabilities defined in Subsection 3.1.1 and 3.1.2. Object size only seems to have a positive effect for large objects (see category *table*), but does not

seem to have a negative effect for small categories such as *cup* or *pottedplant*. The author believes that objects which are describes by a higher number of neighboring pixels are easier detected, because the probability is higher that such objects are not lost in an aggregation step. In such a step the pixels are rather combined and persisted into an object. Apparently also the smaller categories are depicted large enough in this dataset. Geometric complexity is important in the context of non-symmetric categories such as *chair* but not for categories with just a very large surface in comparison to their volume (e.g. category *pottedplant*). In the author's opinion, geometric complex object such as *pottedplant*s still provide a high number of — maybe not neighboring — but close pixels showing the object. In an aggregation step those pixels can be easily grouped together into an object. Geometric complex objects with large,thin structures such as chairs may be depicted on a large number of pixels but they are distributed over a larger area, which makes grouping — and thereby detection — more difficult. Reflectivity and intra-category variability does not have a significant effect on the final accuracy. Due to the high number of images per instance variations are well described and can be easily learned by YOLOv3.

Despite category specific differences systematic differences between the four conditions can be observed. Observable in all augmentation levels is that the lowest accuracy is reached in the *indoor-night* condition. This accuracy variation can be explained by the high number of blurry images in this condition. The highest accuracy is achieved in the *outdoor-sun* condition — where blurry images are an exception. Additionally, because of the direct sun light the images have a very high contrast. The other two conditions reach similar results as the overall evaluation. Observed variations for condition - category - augmentation level combination are random. Hence, the largest deviations from the overall results are due to dataset characteristics and not due to indoor - outdoor variations. The main properties to improve accuracy results of YOLOv3 are sharp images and high contrast.

In summary, the dataset characteristics strongly affect detection results, this includes the best level of data augmentations, the best set of hyperparameters, and category as well as condition specific results. The different results for the three levels of data augmentations demonstrate the importance of selecting a set of augmentation task which matches the data. Additionally, it shows that over-augmentation is possible, if the network never sees original data, then properties can not be generalized to the original data and accuracy results are unsatisfactory. With matching data augmentation significantly better accuracy results can be achieved on this dataset than on MS COCO with YOLOv3. Next 3D accuracy results are evaluated.

6.1.2 3D Data

For 3D object detection evaluation the accuracy results of VoteNet on AR-2/3 are compared to results from the original paper [Qi et al., 2019]. Figure 6.3 visualizes validation mAP for different IoU values. On AR-2/3 by 0.4 and 0.6 higher accuracy results could be achieved. AP_{25} and AP_{50} of AR-2/3 are similar as for SUN RGB-D, but

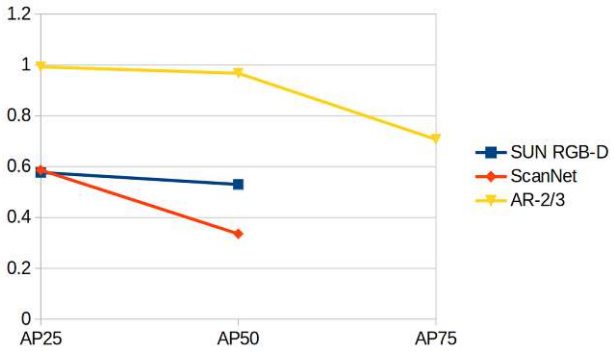


Figure 6.3: Average Precision of best VoteNet validation results on SUN RGB-D, ScenNet, and AR-2/3. VoteNet performs best on AR-2/3.

accuracy drops for AP_{75} though staying above IoU 0.25 and 0.5 results of SUN RGB-D and ScanNet. Due to the additional degrees of freedom in the 3D space achieving good accuracy results for an IoU of 0.75 is quite difficult and therefore often not even evaluated. Hence, those results are satisfactory.

SUN RGB-D has similar specifications as AR-2/3 considering the number of scenes and number of annotated objects, as discussion in Subsection 3.3.2. Therefore, the accuracy improvement from SUN RGB-D to AR-2/3 can be described by two main point cloud properties of AR-2/3. First, a very consistent environment - object setup is used — object is always placed on top of a platform. Second, hardly any additional environment is included in the point cloud (see Subsection 3.3.2). Moreover, a large number of this very consistent scenes is available for training. All of these properties make it easy for the network to learn this specific setup. That is why those better accuracy results can be achieved.

Most tested hyperparameter combinations achieve very good and similar accuracy results (see Figure 5.7). This is another indicator that — partly independent of the hyperparameters — VoteNet is able to learn the training dataset very well or that the dataset is easy to capture for the network. Interestingly for the best run (sorted by AP_{25}) with AR-2/3 nearly the same hyperparameters are used as for SUN RGB-D in the original paper. The only difference is that for this experiment 40,000 instead of 20,000 input points yield the best results. Although the benefit of a larger number of input points could be explained by the additional information provided by more points, it can be argued that the number of input points is of secondary importance. Within the top five runs are also run 1 and run 17 using the exact same hyperparameters as the best run 21 despite the number of input points (20,000 respectively 10,000). Run 1 has even a higher AP_{75} than run 21. Therefore learning rate, batch size and the usage of VoteNet instead of BoxNet seem to be more important hyperparameters.

The comparison between the best VoteNet (run 21) and best BoxNet (run 19) does not show the large accuracy improvement for IoU 0.25 as in [Qi et al., 2019] (improved by

0.05 for SUN RGB-D and 0.13 for ScanNet). In contrast in this experiment for IoU 0.25 the improvement is only 0.0086, for larger IoU values of 0.5 and 0.75 the absolute improvement increases to similar values as in the original paper (0.0328 respectively 0.0751). This effect may be due to the overall high accuracy results, especially for small IoU values, which makes it difficult to achieve high absolute improvements because of the voting step. Hence, the results of this experiment confirm that the voting step improves the accuracy.

An analysis of the category accuracies (see Figure 5.7 and 5.10) shows that the most difficult category for VoteNet is *cup*, for which accuracy decreases rapidly with increasing IoU value. A category with such small extent has not been tested so far as neither SUN RGB-D nor ScanNet provide a comparable category. Because of the small object size and limited measurement capabilities of HoloLens 2 cups may be described by as less as three points in AR-2/3. The small number of points makes them quite difficult to detect and accordingly it is even more difficult to perfectly align a bounding box to achieve high accuracy for large IoU values. For IoU 0.25 and 0.5 the other four categories perform equally well. For category *pottedplant* the best AP_{75} can be achieved. It can be argued that due to their large surface despite their small volume bounding boxes are easy to align. The other three categories show a reasonable accuracy decrease due to the more demanding IoU. Hence, the main object property to consider for high accuracy is the number of object points and their density. Object size is only a limiting factor if the number of object points decreases strongly such as for category *cup*. Small objects with a high number of object points perform well (see *pottedplant*). The number of points depends next to the object size also on the geometric complexity. Geometrically more complex objects with a large surface perform even better than simpler categories such as *monitor*. Object reflectivity does not affect the results in this experiment, as HoloLens two was able to capture all object in all conditions. Also no intra-category effects can be observed. It can be argued that enough samples of all variations are available for training and VoteNet is able to learn all of them.

Evaluating the effect of the four conditions (see Figure 5.8 and 5.9) the low accuracy of category *cup* can be further differentiated. There is a difference of nearly 0.2 between the two indoor and two outdoor conditions. The significantly lower results in the outdoor conditions can be explained by the high number of mapping loss events with category *cup* in those conditions (see Subsection 3.3.2). Because of those events data acquisition was interrupted sometimes even multiple times per instance, the mesh was cleared, and acquisition continued from an empty mesh. That is why *cup* instances are often described only by a very low number of points. Moreover, the least number of point clouds are available for that category and those conditions, which makes it even more difficult for the network to learn such demanding scenes. Similar to the per category evaluation also per condition hardly any variations are observable for low IoU values of 0.25 and 0.5, only for the large IoU of 0.75 more variations appear. Generally, the outdoor condition is more difficult. It can be argued that this is again connected to the more demanding lightning conditions for HoloLens 2's 3D data acquisition in an outdoor environment.

Interestingly the opposite effect can be observed for category *table* and *chair*. For these categories and IoU 0.75 the *indoor-night* condition reaches by far the lowest accuracy. The origin of this effect can not be explained by the available data, further research is required to fully understand the origins of this effect (see Chapter 7).

To sum up, on AR-2/3 VoteNet achieves good accuracy results, which are significantly higher than on SUN RGB-D. Most demanding is the very small *cup* category which performs especially for high IoU values considerably worse than the other categories. This shows that the number of objects points and their density are the most important properties to achieve a good accuracy. Besides mapping issues with the *cup* category in the two outdoor conditions, VoteNet shows no direct effects cause by the different environment / lightning conditions even though it was originally developed for an indoor dataset. In the following the result of both 2D and 3D object detection are compared.

6.2 2D - 3D Comparison

As describes in Subsection 4.3 no previous research is available directly comparing 2D and 3D object detection. For simplicity, the comparison is therefore based on MS COCO metrics leveraging 2D respectively 3D IoU calculation.

To facilitate the comparison between the accuracy results Figure 6.4 summarizes test accuracy values for the best run for the select 2D and 3D algorithm. 2D and 3D *mAP* values are similar, but interestingly the 3D results are slightly better, even though 3D instead of 2D IoU metric is used. In other words VoteNet achieves with the more demanding 3D metric a higher accuracy as YOLOv3 with the simpler 2D metric. For both the accuracy drops with the highest IoU of 0.75. The derived bounding boxes of about a quarter of the detections contain between 50% and 75% of the original bounding box. Hence, the location accuracy is limited. It should be mentioned that a major advantage of YOLOv3 is its fast inference speed, which is a central requirement for the domain of this work, but which also comes at the cost of sightly worse accuracy measures compared to other 2D algorithms such as RetinaNet. Such other algorithms may also reach higher accuracy values in this comparison.

Also per category performs VoteNet generally better than YOLOv3 and accuracy results drop for IoU 0.75 compared to the lower IoU values. The decrease in accuracy with increasing IoU is expected as it is common to most state-of-the-art object detection algorithms in both domains (see e.g. [Aziz et al., 2020], [Rahman et al., 2019]). Besides this general trend there are no similarities between better and worse categories between the 2D and 3D domain. The 3D results are mainly limited by the number of object points and therefore by the category *cup*. Although this is also the smallest category in the 2D domain the *cup* category performs well with YOLOv3. Most demanding is in this domain the geometric complex *chair* category. The low accuracy for IoU 0.75 is in the 2D domain the summary of multiple categories with a lower accuracy. In contrast for the 3D domain the extremely low accuracy results for the *cup* category dominates it. Hence, besides the *cup* category 3D results are significantly better. Next to the

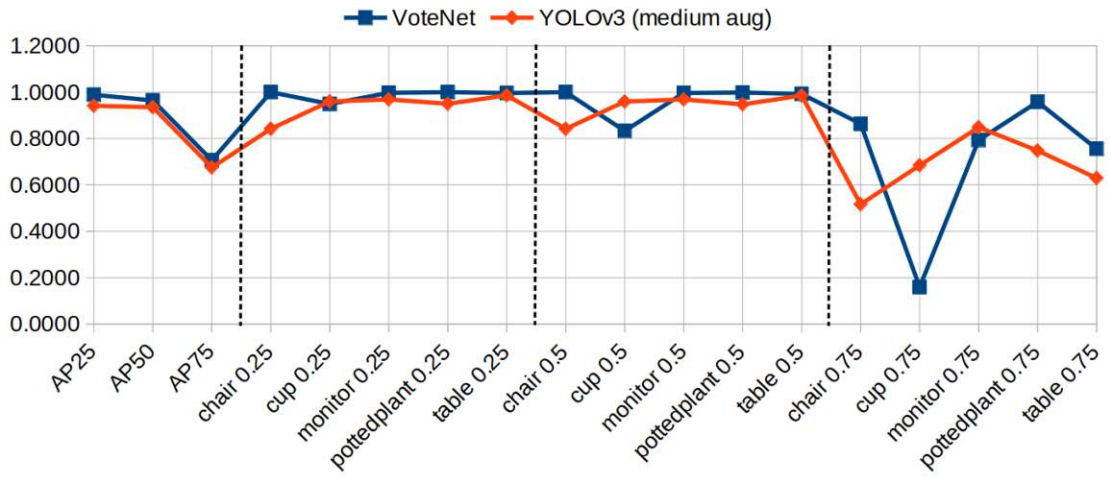


Figure 6.4: Comparison between 2D and 3D test accuracy. Results are generally very similar.

great performance of VoteNet the higher accuracy can be partly attributed to maybe a bit more homogeneous 3D dataset. As no different perspectives of a 3D scenes exist — absolute point locations are measured —, the largest variations in the measured mesh are observable while building the mesh. After creating a first mesh including all object parts, only minor updates are applied, meaning the location of single points change. Hence, a large number of 3D scenes are quite similar, probably even more than in the 2D dataset part.

Condition specific variations are generally independent between the 2D and 3D domain. 2D results are dominated by the best *outdoor-sun* condition with the highest contrast and by the worst *indoor-night* condition with the highest number of blurry images. In contrast 3D results — especially category *cup* — performs better in the two indoor conditions than in the two outdoor conditions. The natural light and in detail direct sun light is no problem for HoloLens 2's RGB camera, but is more problematic for anchoring and creating a spatial mesh especially for very small objects such as cups.

The overall comparison shows that 3D object detection algorithms can compete on an accuracy level with their 2D equivalent. On this dataset the 3D algorithm even achieves a higher accuracy than the 2D one. In addition to the higher accuracy 3D bounding boxes provide with a three dimensional location, extend and rotation more information than 2D bounding boxes. As discussed before this information is required by a number of applications often operating in the three dimensional world.

The main downside of 3D algorithms is the inference time. VoteNet requires with 19.04ms by a factor ten longer than YOLOv3-tiny which needs 1.55ms seconds. YOLOv3-tiny is optimized to achieve fast inference times but even the full equivalent YOLOv3 returns results within 5.16ms seconds only, which is a bit more than a quarter of the time VoteNet

needs. Nevertheless all three algorithms — including VoteNet — achieve the defined real-time goal of 33.3ms on NVIDIA GeForce GTX 1080 based graphic cards. Focusing on AR real-time applications both algorithms are capable of providing fast enough responses to facilitate a smooth real-time experience for a user. Although the difference appears very large it is therefore insignificant and the algorithms can be considered equivalent under this criterion.

In summary, 3D object detection algorithms and in detail VoteNet achieve comparable and better accuracy results then 2D algorithms, here YOLOv3(-tiny). Although 3D algorithms are significantly slower than 2D ones — because of the added complexity — they detect objects in real-time and are therefore also suitable for an AR object detection application. The realization of the experiment — especially the implementation of 3D object detection with AR data — shows that it is possible to combine AR with 3D object detection. The comparison results further demonstrate the advantages of such a combination, resulting 3D bounding boxes do not only provide additional information but also achieve at least a comparable accuracy.

CHAPTER 7

Conclusion and Future Work

In this thesis the accuracy of 2D and 3D object detection algorithms are compared. To enable a fair comparison first a 2D - 3D dataset is acquired. A number of requirements are considered for the dataset concept. Most important are three considerations. First, the provided 2D and 3D data should structurally be as similar as possible to enable a meaningful comparison. Second, generic object categories (*chair*, *cup*, *monitor*, *pottedplant*, *table*) are selected covering a number of different object properties such as size, geometric complexity, and reflectivity. Third, both indoor and outdoor environment and different lightning conditions should be included resulting into four different conditions (*indoor-night*, *indoor-sun*, *outdoor-night*, *outdoor-sun*). To not only allow a fair comparison between 2D and 3D but also between the categories and conditions the same setup is used in all conditions and for all categories. In detail this includes observing each object instance in isolation, placed on top of a platform, with the head mounted AR device HoloLens 2. To optimize data acquisition to the different data characteristics of 2D respectively 3D data one dedicated observation configuration each is developed, defining the exact routes around the platform, walking speed, and measurement frequency. The data acquisition results in approximately 150,000 images and 75,000 3D scenes.

After data acquisition raw data is cleaned from outliers and scenes not containing any object. To use the dataset for object detection with neural networks, objects have to be annotated in the images and 3D scenes — point clouds are used as 3D data representation. 2D data is first manually cleaned. For annotation a small number of images are annotated manually. Those are then used as input for transfer learning on YOLOv3. The resulting weights can in turn be employed to label the complete dataset. Because of the well defined structure of 3D data, objects can be annotated algorithmic. To validate the resulting bounding boxes, their volumes are compared with the bounding box volumes of the same object instance and condition. Outlier are removed. This allows to effectively detect only partly mapped objects and corrupt point clouds automatically. The structured data acquisition and annotation process results in approximately the same number of

annotated images respectively point clouds per category and condition, with the 2D dataset containing roughly twice as many samples as the 3D dataset. All 2D and 3D samples show a homogeneous scenario with a single, mostly iconic object and without any real world context — because of the controlled and static acquisition setup. This homogeneity together with the similarity between consecutive acquisitions, the large number of samples, and the small number of categories makes the new dataset easy to learn. Hence, in comparison to standard datasets detection accuracy is expected to be higher. Further, the equal distribution in combination with the well defined, homogeneous scene content induces that the new dataset is well suitable for academic comparison but is not meant for real-world applications.

To finally compare 2D and 3D object detection algorithms, first one representative algorithm each is selected. For this the test accuracy and inference time on standard datasets as well as the availability of an implementation is considered. YOLOv3 is selected as 2D algorithm VoteNet as 3D algorithm. Both are then trained on the new dataset testing 24 well defined sets of hyperparameters. As initial 2D results are unsatisfactory in total three levels of data augmentation are tested. For the set performing best on validation data test accuracies and inference time are determined and compared to both standard datasets from their respective domain and between each other.

The evaluation of the three levels of data augmentation applied to 2D data shows interesting and unexpected results. Applying most augmentation results in the lowest accuracy, while applying medium or no data augmentation results in very similar and considerably better accuracy. This behavior is explained by three main reasons: First, as previous research shows, the best set of augmentation tasks depends on the used dataset. The set of augmentation tasks selected for medium data augmentation may just better fit the new dataset. Second, it is understood that over-augmentation occurs in the experiment, meaning that each training sample is augmented by a number of task, thereby not showing the network any original sample. The network can not generalize to original samples, which makes it extremely difficult to then detect object in validation and test samples. Third, the very small difference between medium and no augmentation is explained by the large dataset size. Previous research shows that the importance of data augmentation decreases with increasing number of samples. Additionally, it must be mentioned that due to the homogeneity of AR-2/3 the importance and necessity of data augmentation is expect to be low. All these points underline the importance of data augmentation both to improve object detection results, but also its capability to significantly decrease accuracy results. The author argues that in future object detection learning processes either the best set of augmentation task should be evaluated before training, based on the dataset, or the set of applied data augmentation tasks should be considered as hyperparameter.

On the new, simple dataset both 2D and 3D algorithms achieve AP_{25} and AP_{50} above 0.9 and AP_{75} close to 0.7. Because of the dataset's simplicity all results are significantly higher than on standard datasets such as MS COCO or SUN RGB-D. Independent of the absolute accuracy result the main finding is that 2D and 3D mAP results are very

similar, with 3D results being even slightly better than 2D results. Hence, neglecting inference time and only considering accuracy there is no reason to prefer 2D algorithm but rather the opposite. The downside of 3D algorithms in comparison to their 2D equivalents is the inference time. The 3D algorithm VoteNet is with an inference time of 19.04ms by a factor ten slower then the best 2D model leveraging YOLOv3-tiny. But, it still reaches the real-time goal of 33.3ms. 3D algorithms are currently significantly slower then 2D algorithms but they achieve at least comparable accuracy results with the advantage of providing more information — not only a 2D but a 3D bounding box. This finding should be on one hand a starting point to further improve especially the speed of 3D object detection but on the other hand proof that currently existing 3D algorithms already provide good results. They can — and maybe even should — be used more widely in real world applications.

An accuracy evaluation by category and condition shows that 2D and 3D algorithms depend on completely different object and environmental properties and thereby offer advantages and disadvantages in very different areas. The most important property for 3D algorithms is the number and density of object points. The very small *cup* category is by far the worst one. In contrast the also rather small category *pottedplant* performs best. The central difference is that it has a large, easy to map surface and therefore is describe by a large number of dense points. 2D algorithms are less dependent on the object size — though the biggest category *table* performs best — but they are also affect by other properties such as geometric complexity — e.g. category *chair* as non-symmetric object perform worst. Similar observations are made for condition specific result. While for 2D algorithms the *indoor-night* condition with its relatively high number of blurry images is most difficult, for 3D algorithms the outdoor conditions with its high number of mapping loss events is most difficult.

Based on those observations three questions are develop to facilitate the selection of an appropriate object detection method for the AR domain. The following questions assume that 2D and 3D bounding boxes provide sufficient information for the given use case. If 3D information is required the author suggests to directly focus on 3D algorithms.

1. **What is the maximum possible response time?** Although both methods work in real-time, 2D methods are with approximately 1.55ms significantly faster than 3D methods which require 19.04ms.

2. **In which environment do you plan to work?** If you expect direct sunlight 2D methods should be preferred, while in an indoor environment with artificial light only, 3D methods may outperform 2D methods. In case of natural light sources in an indoor environment or no direct sun light in an outdoor environment the methods work equivalently.

3. **Which object categories do you want to detect? / Which 3D mapping resolution is used?** Geometric complex objects with blank like structures — thin in one direction, large in another — are easier to detect with 3D methods. But if

small object categories (with a small surface) are included the mapping resolution has to be high enough to ensure each object is mapped by a large enough number of points. If this cannot be guaranteed 2D methods are significantly better for small categories.

From an scientific point of view the different downsides of both domains should be further optimized. One solution to those problems can be to tackle them separately, e.g. increasing the number of measured points to better describe small objects (3D) or use improved sensors to reduce the number of blurry images (2D). Another more general solution would be to combine 2D and 3D. Both algorithm could benefit from the respective other one and a combined solution is expected to outperform each single algorithm. Future work should, therefore, not only focus on evaluating other 2D (e.g. RetinaNet or YOLOv4) or 3D algorithms (e.g. VoxelNet, SECOND) to validate or falsify the findings of this thesis but also consider algorithms leveraging both 2D and 3D data. Possible candidates are Frustrum PointNet [Qi et al., 2018], SIFRNet [Zhao et al., 2019a], and ImVoteNet [Qi et al., 2020]. To enable the implementation of such combined algorithms 2D and 3D data must be aligned and therefore acquired together. From a technical perspective this requires a combined 2D - 3D client and for the acquisition of such a datatset a new, common acquisition configuration — route, walking speed, measurement frequency. One possibility to realize such a data acquisition would be to map the object in advance with HoloLens 2's spatial mapping capability. During the data acquisition then a roughly complete 3D model is already available and e.g. a similar route as leveraged for 2D data acquisition could be used. Each acquisition is then composed of an image, a point cloud and the current transformation between image and world coordinate system to align image and point cloud.

Further research testing additional models, should also focus on not fully explainable properties of some categories, conditions and category-condition combinations observed in this experiment. The most interesting category would be *chair* for 2D algorithms. As described before category *chair* performs for most hyperparameter combinations significantly worse than all other categories on YOLOv3, although it is a rather large category. Three possible explanations are given in this thesis which should be further validated. First, the least number of samples are available for this category. The importance of this property could be evaluated by capturing and adding additional training samples. Second, *chair* instances are composed of thin structures which may be difficult to distinguish from background. This theory could be proven by testing other categories with similar properties (e.g. pipes or bicycles). Third, chairs are the only not symmetric categories which makes chair detection more demanding. Also this assumption could be analyzed by testing other categories with this property (e.g. animals like dogs).

The most interesting category-condition combination which accuracy deviation could not be explained fully is category *table* and *chair*, condition *indoor-night*. Although 3D data seems to perform better in indoor conditions — due to the more demanding lightning situation in an outdoor environment — those two categories perform worst in

the *indoor-night* condition. One possible explanation is that two different effects act in parallel. In the outdoor conditions the effect of mapping loss events must be considered. In the *indoor-night* condition the RGB autofocus is unreliable and many images are blurry. As for the 3D mesh generation also 2D images are used, blurry images could affect the spatial mesh in the *indoor-night* condition. Depending on which effect is the dominant one — mapping loss for category *cup*, blurry images for category *table* — one condition is better than another one. In this thesis it could not be identified which properties determine which is the dominate effect for a given category. A further analysis including more categories is required to first verify this argument and then identify key properties.

The performed experiment exposes not only the status of 2D and 3D object detection — and left effects of certain categories open — but also demonstrates for the first time how Augmented Reality can be combined with 3D object detection. This could be the starting point for more AR applications leveraging the advantages of 3D object detection. In the experiment only pre-acquired data is used, but the setup used for data acquisition could easily be extended to work in real-time — VoteNet as tested 3D algorithm also achieves the real-time goal. One option to perform real-time 3D object detection using the results of this thesis would be to add an additional endpoint to the server which expects a point cloud, evaluates it on e.g. VoteNet and returns the resulting categories, bounding boxes and optionally scores as JSON. As all 3D coordinates are absolute no additional transformation must be considered. For the 2D case such transformations are required, to visualize bounding box coordinates retrieved in the image coordinate system in the world coordinate system.

In summary, it was possible with the performed experiment to answer the research question on the status of 3D object detection in comparison to its 2D equivalent. State-of-the-art 3D object detection algorithms can achieve comparable results as 3D object detection algorithms even applying the more demanding 3D metrics. Their only downside is the inference time which is still significantly higher. Next steps should focus on accelerating 3D object detection algorithms, further improving accuracy for higher IoU values in both domains, the combination of 2D and 3D data, and the more detailed analysis of some categories and conditions which results can not yet be explained fully. Additionally, this thesis should be seen as proof-of-concept of the combination of Augmented Reality and 3D object detection and as starting point for a stronger integration of these two technologies.

Bibliography

[Ali et al., 2018] Ali, W., Abdelkarim, S., Zidan, M., Zahran, M., and El Sallab, A. (2018). Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, pages 0–0.

[Ammar et al., 2019] Ammar, A., Koubaa, A., Ahmed, M., and Saad, A. (2019). Aerial images processing for car detection using convolutional neural networks: Comparison between faster r-cnn and yolov3. arXiv preprint arXiv:1910.07234.

[Andreopoulos and Tsotsos, 2013] Andreopoulos, A. and Tsotsos, J. K. (2013). 50 years of object recognition: Directions forward. Computer vision and image understanding, 117(8):827–891.

[Arnold et al., 2019] Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., and Mouzakitis, A. (2019). A survey on 3d object detection methods for autonomous driving applications. IEEE Transactions on Intelligent Transportation Systems, 20(10):3782–3795.

[Asgary et al., 2020] Asgary, A., Bonadonna, C., and Frischknecht, C. (2020). Simulation and visualization of volcanic phenomena using microsoft hololens: Case of vulcano island (italy). IEEE Transactions on Engineering Management, 67(3):545–553.

[Attneave and Arnoult, 1956] Attneave, F. and Arnoult, M. D. (1956). The quantitative study of shape and pattern perception. Psychological bulletin, 53(6):452.

[Aziz et al., 2020] Aziz, L., bin Haji Salam, S., and Ayub, S. (2020). Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review. IEEE Access.

[Azuma et al., 2001] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. (2001). Recent advances in augmented reality. IEEE computer graphics and applications, 21(6):34–47.

[Azuma, 1997] Azuma, R. T. (1997). A survey of augmented reality. Presence: Teleoperators & Virtual Environments, 6(4):355–385.

[Bacca Acosta et al., 2014] Bacca Acosta, J. L., Baldiris Navarro, S. M., Fabregat Gesa, R., Graf, S., et al. (2014). Augmented reality trends in education: a systematic review of research and applications. Journal of Educational Technology and Society, 2014, vol. 17, núm. 4, p. 133-149.

[Beltrán et al., 2018] Beltrán, J., Guindel, C., Moreno, F. M., Cruzado, D., Garcia, F., and De La Escalera, A. (2018). Birdnet: a 3d object detection framework from lidar information. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 3517–3523. IEEE.

[Bengio, 2009] Bengio, Y. (2009). Learning deep architectures for AI. Now Publishers Inc.

[Bengio et al., 2017] Bengio, Y., Goodfellow, I., and Courville, A. (2017). Deep learning, volume 1. MIT press Massachusetts, USA:.

[Benjdira et al., 2019] Benjdira, B., Khursheed, T., Koubaa, A., Ammar, A., and Ouni, K. (2019). Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3. In 2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS), pages 1–6. IEEE.

[Billinghurst et al., 2015] Billinghurst, M., Clark, A., and Lee, G. (2015). A survey of augmented reality.

[Birkfellner et al., 2002] Birkfellner, W., Figl, M., Huber, K., Watzinger, F., Wanschitz, F., Hummel, J., Hanel, R., Greimel, W., Homolka, P., Ewers, R., et al. (2002). A head-mounted operating binocular for augmented reality visualization in medicine-design and initial evaluation. IEEE Transactions on Medical Imaging, 21(8):991–997.

[Bochkovskiy et al., 2020] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[Bottou and Bousquet, 2011] Bottou, L. and Bousquet, O. (2011). 13 the tradeoffs of large-scale learning. Optimization for machine learning, page 351.

[Breuel, 2015] Breuel, T. M. (2015). The effects of hyperparameters on sgd training of neural networks. arXiv preprint arXiv:1508.02788.

[Caesar et al., 2020] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuscenes: A multimodal dataset for autonomous driving. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 11621–11631.

[Chen et al., 2016] Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., and Urtasun, R. (2016). Monocular 3d object detection for autonomous driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2147–2156.

[Chen et al., 2017] Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 1907–1915.

[Chen et al., 2021] Chen, Y., Ma, H., Li, X., and Luo, X. (2021). S-votenet: Deep hough voting with spherical proposal for 3d object detection. In 2020 25th International Conference on Pattern Recognition (ICPR), pages 5161–5167. IEEE.

[Chu et al., 2020] Chu, C.-H., Liao, C.-J., and Lin, S.-C. (2020). Comparing augmented reality-assisted assembly functions—a case study on dougong structure. Applied Sciences, 10(10):3383.

[Claesen and De Moor, 2015] Claesen, M. and De Moor, B. (2015). Hyperparameter search in machine learning. arXiv preprint arXiv:1502.02127.

[Cubuk et al., 2018] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501.

[Cubuk et al., 2020] Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 702–703.

[Dai et al., 2016] Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. arXiv preprint arXiv:1605.06409.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), volume 1, pages 886–893. Ieee.

[de Souza Cardoso et al., 2020] de Souza Cardoso, L. F., Mariano, F. C. M. Q., and Zorzal, E. R. (2020). A survey of industrial augmented reality. Computers & Industrial Engineering, 139:106159.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee.

[Deng and Yu, 2014] Deng, L. and Yu, D. (2014). Deep learning: methods and applications. Foundations and trends in signal processing, 7(3–4):197–387.

[Dogo et al., 2018] Dogo, E., Afolabi, O., Nwulu, N., Twala, B., and Aigbavboa, C. (2018). A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. In 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), pages 92–99. IEEE.

[Drost et al., 2017] Drost, B., Ulrich, M., Bergmann, P., Hartinger, P., and Steger, C. (2017). Introducing mvtec itodd-a dataset for 3d object recognition in industry. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 2200–2208.

[Dünser, 2008] Dünser, A. (2008). Supporting low ability readers with interactive augmented reality. Annual review of cybertherapy and telemedicine, 6(1):39–46.

[Eckert et al., 2018] Eckert, M., Blex, M., Friedrich, C. M., et al. (2018). Object detection featuring 3d audio localization for microsoft hololens. In Proc. 11th Int. Joint Conf. on Biomedical Engineering Systems and Technologies, volume 5, pages 555–561.

[Engelcke et al., 2017] Engelcke, M., Rao, D., Wang, D. Z., Tong, C. H., and Posner, I. (2017). Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 1355–1361. IEEE.

[Everingham et al., 2012] Everingham, M., Van Gool, L., Williams, C., Winn, J., and Zisserman, A. (2012). The pascal visual object classes challenge 2012 (voc2012) results (2012). In URL http://www. pascal-network. org/challenges/VOC/voc2011/workshop/index. html.

[Everingham et al., 2007] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2007). The pascal visual object classes challenge 2007 (voc2007) results.

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2):303–338.

[Farasin et al., 2020] Farasin, A., Peciarolo, F., Grangetto, M., Gianaria, E., and Garza, P. (2020). Real-time object detection and tracking in mixed reality using microsoft hololens. In 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2020, volume 4, pages 165–172. SciTePress.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395.

[Fischler and Elschlager, 1973] Fischler, M. A. and Elschlager, R. A. (1973). The representation and matching of pictorial structures. IEEE Transactions on computers, 100(1):67–92.

[Fraga-Lamas et al., 2018] Fraga-Lamas, P., Fernandez-Carames, T. M., Blanco-Novoa, O., and Vilar-Montesinos, M. A. (2018). A review on industrial augmented reality systems for the industry 4.0 shipyard. Ieee Access, 6:13358–13375.

[Friederich and Zschech, 2020] Friederich, J. and Zschech, P. (2020). Review and systematization of solutions for 3d object detection. In Proceedings of the 15th International Conference on Wirtschaftsinformatik (WI), pages 1699–1711.

[Funk et al., 2017] Funk, M., Bächler, A., Bächler, L., Kosch, T., Heidenreich, T., and Schmidt, A. (2017). Working with augmented reality? a long-term analysis of in-situ instructions at the assembly workplace. In Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments, pages 222–229.

[Gaidon et al., 2016] Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4340–4349.

[Geiger et al., 2012] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 3354–3361. IEEE.

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

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587.

[Girshick et al., 2015] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2015). Region-based convolutional networks for accurate object detection and segmentation. IEEE transactions on pattern analysis and machine intelligence, 38(1):142–158.

[Glocker et al., 2014] Glocker, B., Shotton, J., Criminisi, A., and Izadi, S. (2014). Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding. IEEE transactions on visualization and computer graphics, 21(5):571–583.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, pages 315–323. JMLR Workshop and Conference Proceedings.

[Hariharan et al., 2014] Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2014). Simultaneous detection and segmentation. In European Conference on Computer Vision, pages 297–312. Springer.

[Harris et al., 1988] Harris, C. G., Stephens, M., et al. (1988). A combined corner and edge detector. In Alvey vision conference, volume 15, pages 10–5244. Citeseer.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 37(9):1904–1916.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

[Helmstaedter et al., 2013] Helmstaedter, M., Briggman, K. L., Turaga, S. C., Jain, V., Seung, H. S., and Denk, W. (2013). Connectomic reconstruction of the inner plexiform layer in the mouse retina. Nature, 500(7461):168–174.

[Hinton et al., 2012] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine, 29(6):82–97.

[Hodan et al., 2017] Hodan, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., and Zabulis, X. (2017). T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 880–888. IEEE.

[Howard et al., 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[Huang et al., 2018] Huang, R., Pedoeem, J., and Chen, C. (2018). Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers. In 2018 IEEE International Conference on Big Data (Big Data), pages 2503–2510. IEEE.

[Hübner et al., 2020] Hübner, P., Clintworth, K., Liu, Q., Weinmann, M., and Wursthorn, S. (2020). Evaluation of hololens tracking and depth sensing for indoor mapping applications. Sensors, 20(4):1021.

[Hübner et al., 2019] Hübner, P., Landgraf, S., Weinmann, M., and Wursthorn, S. (2019). Evaluation of the microsoft hololens for the mapping of indoor building environments. Proceedings of the Dreiländertagung der DGPF, der OVG und der SGPF, Vienna, Austria, pages 20–22.

[Izadi et al., 2011] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al. (2011). Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In Proceedings of the 24th annual ACM symposium on User interface software and technology, pages 559–568.

[Janoch et al., 2013] Janoch, A., Karayev, S., Jia, Y., Barron, J. T., Fritz, M., Saenko, K., and Darrell, T. (2013). A category-level 3d object dataset: Putting the kinect to work. In Consumer depth cameras for computer vision, pages 141–165. Springer.

[Kang et al., 2017] Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., et al. (2017). T-cnn: Tubelets with convolutional neural networks for object detection from videos. IEEE Transactions on Circuits and Systems for Video Technology, 28(10):2896–2907.

[Karpathy and Fei-Fei, 2015] Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3128–3137.

[Kästner et al., 2021] Kästner, L., Eversberg, L., Mursa, M., and Lambrecht, J. (2021). Integrative object and pose to task detection for an augmented-reality-based human assistance system using neural networks. In 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE), pages 332–337. IEEE.

[Khan et al., 2015] Khan, D., Ullah, S., and Rabbi, I. (2015). Factors affecting the design and tracking of artoolkit markers. Computer Standards & Interfaces, 41:56–66.

[Khoshelham et al., 2019] Khoshelham, K., Tran, H., and Acharya, D. (2019). Indoor mapping eyewear: geometric evaluation of spatial mapping capability of hololens.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[Krasin et al., 2017] Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Veit, A., et al. (2017). Openimages: A public dataset for large-scale multi-label and multi-class image classification. Dataset available from https://github.com/openimages, 2(3):18.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25:1097–1105.

[Ku et al., 2018] Ku, J., Mozifian, M., Lee, J., Harakeh, A., and Waslander, S. L. (2018). Joint 3d proposal generation and object detection from view aggregation. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1–8. IEEE.

[Kukačka et al., 2017] Kukačka, J., Golkov, V., and Cremers, D. (2017). Regularization for deep learning: A taxonomy. arXiv preprint arXiv:1710.10686.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. nature, 521(7553):436–444.

[Lee et al., 2019] Lee, S., Lee, S., Lee, K., and Ko, J. G. (2019). Deepmobilear: A mobile augmented reality application with integrated visual slam and object detection. In SIGGRAPH Asia 2019 Posters, pages 1–2.

[Li, 2017] Li, B. (2017). 3d fully convolutional network for vehicle detection in point cloud. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1513–1518. IEEE.

[Li et al., 2016] Li, B., Zhang, T., and Xia, T. (2016). Vehicle detection from 3d lidar using fully convolutional network. arXiv preprint arXiv:1608.07916.

[Li et al., 2020] Li, X., Tian, Y., Zhang, F., Quan, S., and Xu, Y. (2020). Object detection in the context of mobile augmented reality. In 2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pages 156–163. IEEE.

[Lim et al., 2019] Lim, S., Kim, I., Kim, T., Kim, C., and Kim, S. (2019). Fast autoaugment. arXiv preprint arXiv:1905.00397.

[Lin et al., 2013] Lin, M., Chen, Q., and Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400.

[Lin et al., 2017a] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2117–2125.

[Lin et al., 2017b] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision, pages 2980–2988.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In European conference on computer vision, pages 740–755. Springer.

[Liu et al., 2020] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. International journal of computer vision, 128(2):261–318.

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In European conference on computer vision, pages 21–37. Springer.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In Proceedings of the seventh IEEE international conference on computer vision, volume 2, pages 1150–1157. Ieee.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2):91–110.

[Ma et al., 2015] Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., and Svetnik, V. (2015). Deep neural nets as a method for quantitative structure–activity relationships. Journal of chemical information and modeling, 55(2):263–274.

[Mahurkar, 2018] Mahurkar, S. (2018). Integrating yolo object detection with augmented reality for ios apps. In 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), pages 585–589. IEEE.

[McGill et al., 1978] McGill, R., Tukey, J. W., and Larsen, W. A. (1978). Variations of box plots. The American Statistician, 32(1):12–16.

[Microsoft, 2021a] Microsoft (2021a). HoloLens (1st gen) hardware. https://docs.microsoft.com/en-us/hololens/hololens1-hardware. [Online; accessed 25-March-2021].

[Microsoft, 2021b] Microsoft (2021b). HoloLens 2. https://www.microsoft.com/en-us/hololens/hardware. [Online; accessed 23-March-2021].

[Mikołajczyk and Grochowski, 2018] Mikołajczyk, A. and Grochowski, M. (2018). Data augmentation for improving deep learning in image classification problem. In 2018 international interdisciplinary PhD workshop (IIPhDW). IEEE.

[Minemura et al., 2018] Minemura, K., Liau, H., Monrroy, A., and Kato, S. (2018). Lmnet: Real-time multiclass object detection on cpu using 3d lidar. In 2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), pages 28–34. IEEE.

[Mousavian et al., 2017] Mousavian, A., Anguelov, D., Flynn, J., and Kosecka, J. (2017). 3d bounding box estimation using deep learning and geometry. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7074–7082.

[Mundy, 2006] Mundy, J. L. (2006). Object recognition in the geometric era: A retrospective. Toward category-level object recognition, pages 3–28.

[Murase and Nayar, 1995] Murase, H. and Nayar, S. K. (1995). Visual learning and recognition of 3-d objects from appearance. International journal of computer vision, 14(1):5–24.

[Naseer et al., 2018] Naseer, M., Khan, S., and Porikli, F. (2018). Indoor scene understanding in 2.5/3d for autonomous agents: A survey. IEEE Access, 7:1859–1887.

[Nießner et al., 2013] Nießner, M., Zollhöfer, M., Izadi, S., and Stamminger, M. (2013). Real-time 3d reconstruction at scale using voxel hashing. ACM Transactions on Graphics (ToG), 32(6):1–11.

[Nishihara and Okamoto, 2015] Nishihara, A. and Okamoto, J. (2015). Object recognition in assembly assisted by augmented reality system. In 2015 SAI Intelligent Systems Conference (IntelliSys), pages 400–407. IEEE.

[Padilla et al., 2021] Padilla, R., Passos, W. L., Dias, T. L., Netto, S. L., and da Silva, E. A. (2021). A comparative analysis of object detection metrics with a companion open-source toolkit. Electronics, 10(3):279.

[Park et al., 2020a] Park, B. J., Hunt, S. J., Nadolski, G. J., and Gade, T. P. (2020a). 3d augmented reality-assisted ct-guided interventions: System design and preclinical trial on an abdominal phantom using hololens 2. arXiv preprint arXiv:2005.09146.

[Park et al., 2020b] Park, B. J., Hunt, S. J., Nadolski, G. J., and Gade, T. P. (2020b). 3d augmented reality-assisted ct-guided interventions: System design and preclinical trial on an abdominal phantom using hololens 2. CoRR, abs/2005.09146.

[Park et al., 2020c] Park, K.-B., Kim, M., Choi, S. H., and Lee, J. Y. (2020c). Deep learning-based smart task assistance in wearable augmented reality. Robotics and Computer-Integrated Manufacturing, 63:101887.

[Peng et al., 2018] Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., and Sun, J. (2018). Megdet: A large mini-batch object detector. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6181–6189.

[Perez and Wang, 2017] Perez, L. and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621.

[Puljiz et al., 2020] Puljiz, D., Krebs, F., Bösing, F., and Hein, B. (2020). What the hololens maps is your workspace: Fast mapping and set-up of robot cells via head mounted displays and augmented reality. arXiv preprint arXiv:2005.12651.

[Qi et al., 2020] Qi, C. R., Chen, X., Litany, O., and Guibas, L. J. (2020). Imvotenet: Boosting 3d object detection in point clouds with image votes. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 4404–4413.

[Qi et al., 2019] Qi, C. R., Litany, O., He, K., and Guibas, L. J. (2019). Deep hough voting for 3d object detection in point clouds. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 9277–9286.

[Qi et al., 2018] Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. (2018). Frustum pointnets for 3d object detection from rgb-d data. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 918–927.

[Qi et al., 2017a] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 652–660.

[Qi et al., 2017b] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413.

[Radkowski, 2016] Radkowski, R. (2016). Object tracking with a range camera for augmented reality assembly assistance. Journal of Computing and Information Science in Engineering, 16(1).

[Rahman et al., 2019] Rahman, M. M., Tan, Y., Xue, J., and Lu, K. (2019). Recent advances in 3d object detection in the era of deep neural networks: A survey. IEEE Transactions on Image Processing, 29:2947–2962.

[Rajendran et al., 2019] Rajendran, S. P., Shine, L., Pradeep, R., and Vijayaraghavan, S. (2019). Real-time traffic sign recognition using yolov3 based detector. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pages 1–7. IEEE.

[Rao et al., 2017] Rao, J., Qiao, Y., Ren, F., Wang, J., and Du, Q. (2017). A mobile outdoor augmented reality method combining deep learning object detection and spatial relationships for geovisualization. Sensors, 17(9):1951.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 779–788.

[Redmon and Farhadi, 2017] Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7263–7271.

[Redmon and Farhadi, 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497.

[Ren and Sudderth, 2016] Ren, Z. and Sudderth, E. B. (2016). Three-dimensional object detection and layout prediction using clouds of oriented gradients. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1525–1533.

[Ren and Sudderth, 2018] Ren, Z. and Sudderth, E. B. (2018). 3d object detection with latent support surfaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 937–946.

[Roberts, 1963] Roberts, L. G. (1963). Machine perception of three-dimensional solids. PhD thesis, Massachusetts Institute of Technology.

[Sainath et al., 2013] Sainath, T. N., Mohamed, A.-r., Kingsbury, B., and Ramabhadran, B. (2013). Deep convolutional neural networks for lvcsr. In 2013 IEEE international conference on acoustics, speech and signal processing, pages 8614–8618. IEEE.

[Schmid and Mohr, 1997] Schmid, C. and Mohr, R. (1997). Local grayvalue invariants for image retrieval. IEEE transactions on pattern analysis and machine intelligence, 19(5):530–535.

[Schneider et al., 2021] Schneider, M., Kunz, C., Pal'a, A., Wirtz, C. R., Mathis-Ullrich, F., and Hlaváč, M. (2021). Augmented reality–assisted ventriculostomy. Neurosurgical Focus, 50(1):E16.

[Serrano Vergel et al., 2020] Serrano Vergel, R., Morillo Tena, P., Casas Yrurzum, S., and Cruz-Neira, C. (2020). A comparative evaluation of a virtual reality table and a hololens-based augmented reality system for anatomy training. IEEE Transactions on Human-Machine Systems, 50(4):337–348.

[Sharma and Mir, 2020] Sharma, V. and Mir, R. N. (2020). A comprehensive and systematic look up into deep learning based object detection techniques: A review. Computer Science Review, 38:100301.

[Shen, 2019] Shen, X. (2019). A survey of object classification and detection based on 2d/3d data. arXiv preprint arXiv:1905.12683.

[Shin et al., 2019] Shin, K., Kwon, Y. P., and Tomizuka, M. (2019). Roarnet: A robust 3d object detection based on region approximation refinement. In 2019 IEEE Intelligent Vehicles Symposium (IV), pages 2510–2515. IEEE.

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(1):1–48.

[Silberman et al., 2012] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In European conference on computer vision, pages 746–760. Springer.

[Simony et al., 2018] Simony, M., Milzy, S., Amendey, K., and Gross, H.-M. (2018). Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, pages 0–0.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[Singh et al., 2019] Singh, R. D., Mittal, A., and Bhatia, R. K. (2019). 3d convolutional neural network for object recognition: a review. Multimedia Tools and Applications, 78(12):15951–15995.

[Song et al., 2015] Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 567–576.

[Song and Xiao, 2014] Song, S. and Xiao, J. (2014). Sliding shapes for 3d object detection in depth images. In European conference on computer vision, pages 634–651. Springer.

[Song and Xiao, 2016] Song, S. and Xiao, J. (2016). Deep sliding shapes for amodal 3d object detection in rgb-d images. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 808–816.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958.

[Strubell et al., 2019] Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:1906.02243.

[Sultana et al., 2019] Sultana, F., Sufian, A., and Dutta, P. (2019). A review of object detection models based on convolutional neural network. arXiv preprint arXiv:1905.01614.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9.

[Tan et al., 2018] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning. In International conference on artificial neural networks, pages 270–279. Springer.

[Terry, 2019] Terry, E. (2019). Silicon at the heart of hololens 2. In 2019 IEEE Hot Chips 31 Symposium (HCS), pages 1–26. IEEE Computer Society.

[Thipsanthia et al., 2019] Thipsanthia, P., Chamchong, R., and Songram, P. (2019). Road sign detection and recognition of thai traffic based on yolov3. In International Conference on Multi-disciplinary Trends in Artificial Intelligence, pages 271–279. Springer.

[Thomas and David, 1992] Thomas, P. C. and David, W. (1992). Augmented reality: An application of heads-up display technology to manual manufacturing processes. In Hawaii international conference on system sciences, pages 659–669.

[Tremblay et al., 2018] Tremblay, J., To, T., and Birchfield, S. (2018). Falling things: A synthetic dataset for 3d object detection and pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 2038–2041.

[Ungureanu et al., 2020] Ungureanu, D., Bogo, F., Galliani, S., Sama, P., Duan, X., Meekhof, C., Stühmer, J., Cashman, T. J., Tekin, B., Schönberger, J. L., et al. (2020). Hololens 2 research mode as a tool for computer vision research. arXiv preprint arXiv:2008.11239.

[Valiati and Menotti, 2019] Valiati, G. R. and Menotti, D. (2019). Detecting pedestrians with yolov3 and semantic segmentation infusion. In 2019 International Conference on Systems, Signals and Image Processing (IWSSIP), pages 95–100. IEEE.

[Van Krevelen and Poelman, 2010] Van Krevelen, D. and Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. International journal of virtual reality, 9(2):1–20.

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, volume 1, pages I–I. IEEE.

[Viola and Jones, 2004] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. International journal of computer vision, 57(2):137–154.

[Wang and Posner, 2015] Wang, D. Z. and Posner, I. (2015). Voting for voting in online point cloud object detection. In Robotics: Science and Systems, volume 1, pages 10–15607. Rome, Italy.

[Wang et al., 2013] Wang, Z., Ong, S., and Nee, A. (2013). Augmented reality aided interactive manual assembly design. The International Journal of Advanced Manufacturing Technology, 69(5):1311–1321.

[Xiang et al., 2015] Xiang, Y., Choi, W., Lin, Y., and Savarese, S. (2015). Data-driven 3d voxel patterns for object category recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1903–1911.

[Xiao et al., 2013] Xiao, J., Owens, A., and Torralba, A. (2013). Sun3d: A database of big spaces reconstructed using sfm and object labels. In Proceedings of the IEEE international conference on computer vision, pages 1625–1632.

[Xiao et al., 2020] Xiao, Y., Tian, Z., Yu, J., Zhang, Y., Liu, S., Du, S., and Lan, X. (2020). A review of object detection based on deep learning. Multimedia Tools and Applications, 79(33):23729–23791.

[Xie et al., 2020] Xie, Q., Lai, Y.-K., Wu, J., Wang, Z., Zhang, Y., Xu, K., and Wang, J. (2020). Mlcvnet: Multi-level context votenet for 3d object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10447–10456.

[Xu et al., 2018] Xu, D., Anguelov, D., and Jain, A. (2018). Pointfusion: Deep sensor fusion for 3d bounding box estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 244–253.

[Yan et al., 2018] Yan, Y., Mao, Y., and Li, B. (2018). Second: Sparsely embedded convolutional detection. Sensors, 18(10):3337.

[Yang et al., 2018] Yang, B., Luo, W., and Urtasun, R. (2018). Pixor: Real-time 3d object detection from point clouds. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 7652–7660.

[Yang et al., 2019] Yang, S., Zhang, J., Bo, C., Wang, M., and Chen, L. (2019). Fast vehicle logo detection in complex scenes. Optics & Laser Technology, 110:196–201.

[Yi et al., 2016] Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. (2016). A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (ToG), 35(6):1–12.

[Yu et al., 2017] Yu, S.-L., Westfechtel, T., Hamada, R., Ohno, K., and Tadokoro, S. (2017). Vehicle detection and localization on bird's eye view elevation images using convolutional neural network. In 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), pages 102–109. IEEE.

[Zeller, 2018] Zeller, M. (2018). Spatial mapping. https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping. [Online; accessed 18-January-2021].

[Zhang et al., 2018] Zhang, W., Yang, G., Lin, Y., Ji, C., and Gupta, M. M. (2018). On definition of deep learning. In 2018 World automation congress (WAC), pages 1–5. IEEE.

[Zhang et al., 2019] Zhang, Y., Li, D., Wang, H., and Yang, Z.-H. (2019). Application of mixed reality based on hololens in nuclear power engineering. In International Symposium on Software Reliability, Industrial Safety, Cyber Security and Physical Protection for Nuclear Power Plant, pages 9–20. Springer.

[Zhao et al., 2019a] Zhao, X., Liu, Z., Hu, R., and Huang, K. (2019a). 3d object detection using scale invariant and feature reweighting networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 9267–9274.

[Zhao et al., 2019b] Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. (2019b). Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems, 30(11):3212–3232.

[Zhong et al., 2020] Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2020). Random erasing data augmentation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 13001–13008.

[Zhou and Tuzel, 2018] Zhou, Y. and Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4490–4499.

[Zoph et al., 2020] Zoph, B., Cubuk, E. D., Ghiasi, G., Lin, T.-Y., Shlens, J., and Le, Q. V. (2020). Learning data augmentation strategies for object detection. In European Conference on Computer Vision, pages 566–583. Springer.

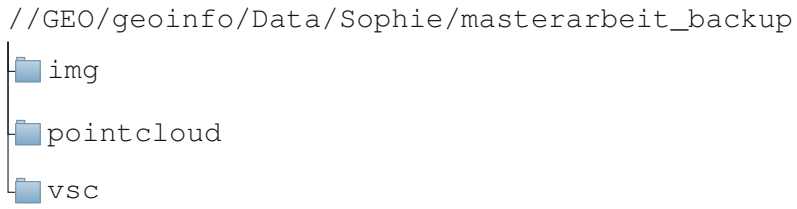
[Zou et al., 2019] Zou, Z., Shi, Z., Guo, Y., and Ye, J. (2019). Object detection in 20 years: A survey. arXiv preprint arXiv:1905.05055.

Appendix

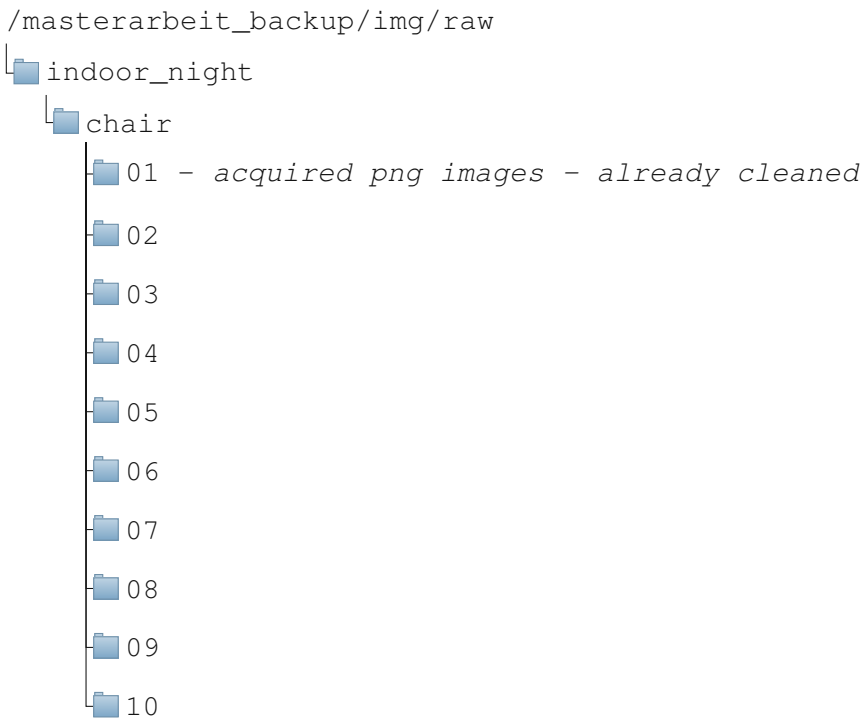
Dataset Structure

A large number of data sample are acquired, including raw images, raw point cloud data, auxiliary data, and intermediate as well as final results. Subsequently, important folders are explained.

There is a main separation into images, pointclouds, and things generated by or needed for the VSC setup. Additionally the configuration and documentation file `train_run_params.ods` is available in the main folder. For easier usage with Microsoft Excel this file is also provided as `.xlsx`. Next, first image data, then pointcloud and finally vsc files are explained.



The most important image data are the acquired and cleaned images (see `masterarbeit_backup/img/raw`). Data is organized per condition (*indoor-night*, *indoor-sun*, *outdoor-night*, *outdoor-sun*), per category (*chair*, *cup*, *monitor*, *pottedplant*, *table*) and per object instance (from 01 to 10). Each folder contains png images. Below an example of of the condition *indoor-night* and category *chair* is given.

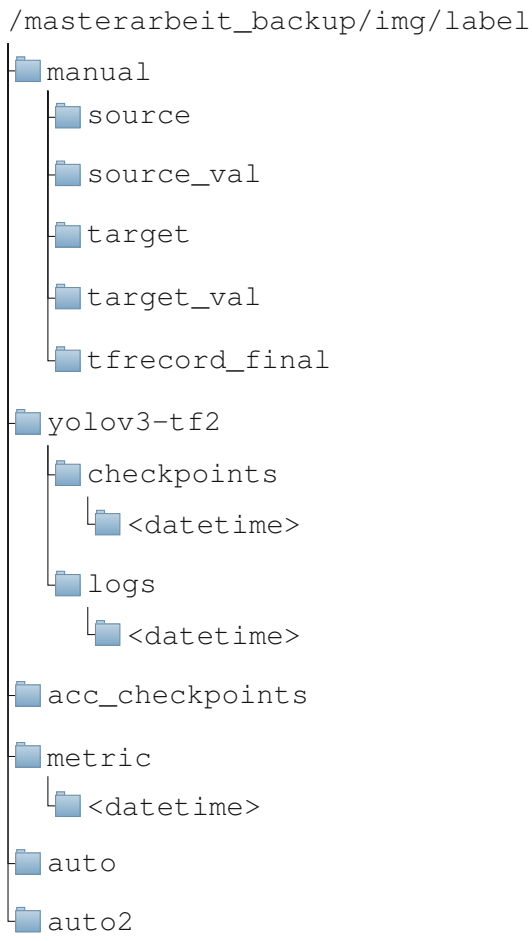


As described in Subsection 3.1.5 the 2D labeling process includes multiple step. Data generated throughout this process is available under `masterarbeit_backup/img/label` (see folder structure below). It is separated into multiple folders: `manual` includes input and output data of Microsoft VOTT, used for manually labeling images. In detail, the two `source` and `source_val` sub-folder store raw png files — the first one is composed of 5000 train samples and the latter of 1000 validation samples. The two `target` and `target_val` sub-folders provide Microsoft VOTT json descriptions per image, the exported Pascal VOC description and the final tfrecords per split.

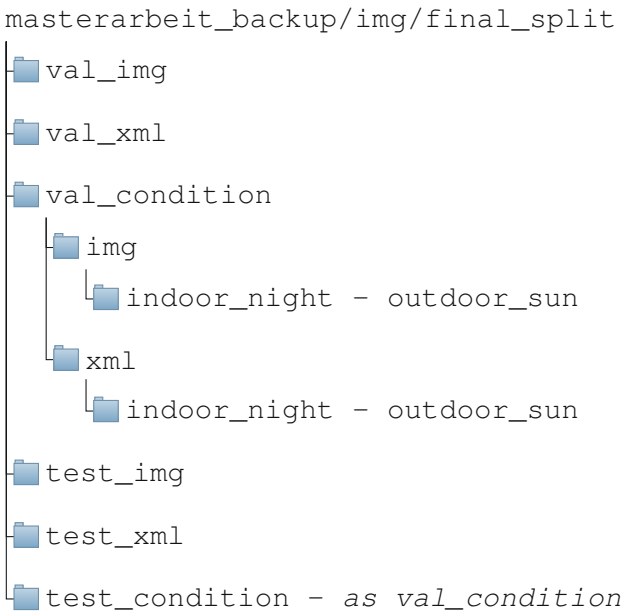
Based on the manually labeled subset, YOLOv3 is trained with different sets of hyperparameters. Each set of hyperparameters is named by the execution datetime (see `train_run_params.ods` for mapping). Resulting checkpoint files and logs are available under `masterarbeit_backup/img/label/yolov3-tf2`. Checkpoints for each epoch are stored. Those are then used to calculate accuracy results. For this the checkpoints of the final epoch are copied to `masterarbeit_backup/img/label/acc_checkpoints` and the validation dataset is labeled with each checkpoint file. Results per datetime in YOLO format are available under `masterarbeit_backup/img/label/metric/<datetime>`. Those labeled files are used as input for `review_object_detection_metrics` to determine the final validation accuracy. Results are available in the main `train_run_params.ods`. The best run is then used to label all images.

The folder `auto` and `auto2` provide the final output of the labeling process of all images. This includes Pascal VOC xml files and tfrecords. The difference between the two folders is that `auto` includes all accepted images — which means the train split contains 112,514

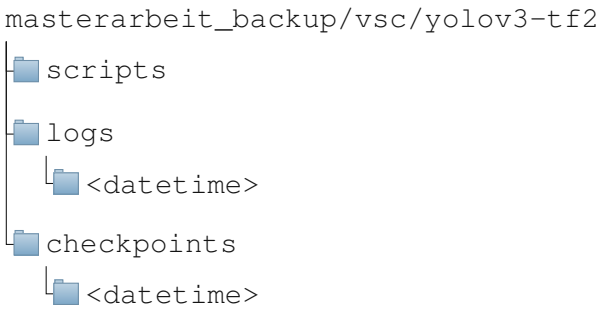
samples — and `auto2` includes only 112,500 images. The latter is required for distributed training of YOLOv3 with Tensorflow 2.1 and Tensorflow 2.3. For distributed training the number of training and validation samples must divide evenly by the number of used GPUs. Hence, the latter dataset is used for hyperparameter sets which can only be trained on multiple GPUs.



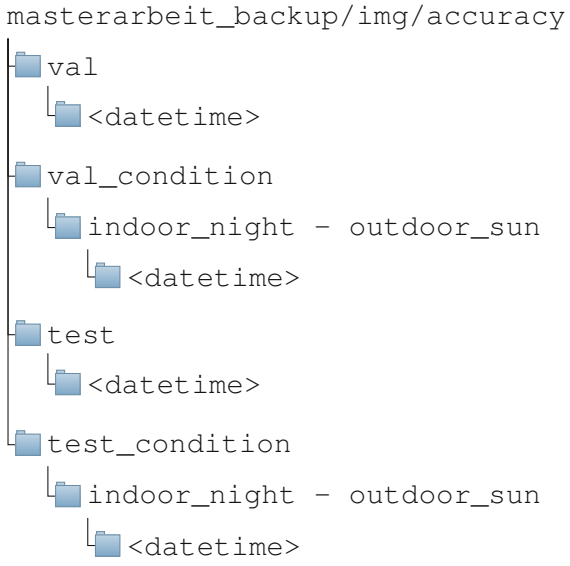
The labeled data is then split into train, validation, and test set. Validation respectively test images and Pascal VOC xml files are available in two formats, either including all conditions or per condition (see `masterarbeit_backup/img/final_split`). Those are required to latter calculate accuracy metrics on the respective (sub-) dataset.



Leveraging the fully labeled dataset YOLOv3 is trained again with multiple sets of hyperparameters. The object detection experiment is performed on VSC3. Scripts, logs and checkpoints are stored under `masterarbeit_backup/vsc/yolov3-tf2`. Similar to the training for the labeling process, hyperparameter sets are again named by there execution datetime. The mapping can again be found in `train_run_params.ods`.

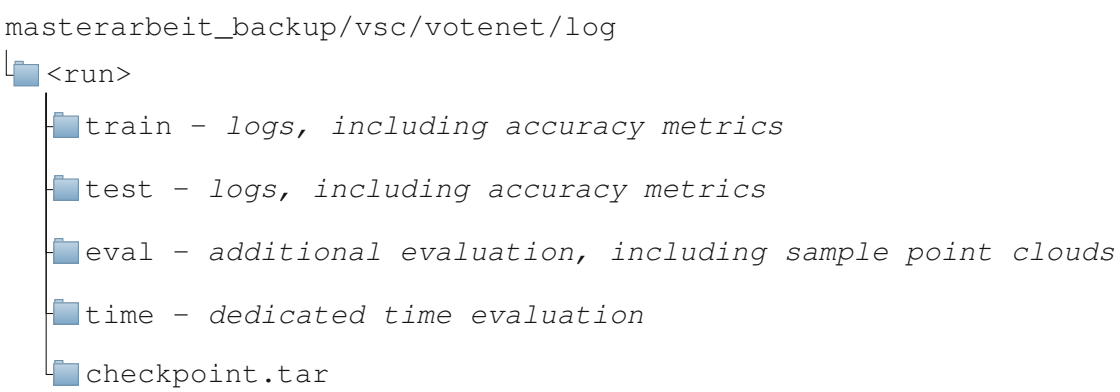


Per run accuracy results are calculated. For this validation respectively test set is labeled. Detected objects are stored per images and per run in YOLO format (see `masterarbeit_backup/img/accuracy`) to be used as input for `review_object_detection_metrics`. Validation accuracy is calculated for all runs, test accuracy only for the best run per augmentation level — three in total. Final results are summarized in `train_run_params.ods`.



After detailing the 2D data structure the organization of the 3D data is described subsequently. Acquired point clouds are available under `masterarbeit_backup/pointcloud/raw`, equivalent to raw 2D data they are organized per condition, category and object instance. The 3D labeling process includes less steps than the 2D one so also the `masterarbeit_backup/pointcloud/label` folder is simpler. It is composed of one .npy file per accepted point cloud. It describes detected objects and their bounding boxes in the format as expected by VoteNet. Folders are again structured by condition, category and instance. The final input for votenet (see `masterarbeit_backup/pointcloud/votenet_input`) includes the point clouds (.npz), bounding boxes (.npy), and votes (.npy) in the format as expected by Votenet. Data is separated into train, val, and test split. For condition specific evaluation validation and test split are also available per condition (see `masterarbeit_backup/pointcloud/condition_split`).

Training results of the object detection experiment realized at VSC3 are available under `masterarbeit_backup/vsc/votenet/log`. Each set of hyperparameters is assigned a so called *run number* (for mapping see `train_run_params.ods`). Per run training logs, test logs, evaluation metrics, and the checkpoint file of the final epoch are available. As accuracy metrics are calculated directly no additional setup — and no additional folders — are required.



Code repositories

Throughout this thesis five main code repositories are created or edited. Subsequently a general description for each repository is given. Detailed descriptions and setup instructions are available in the `README.md` in each repository. The latter three repositories are forks from other repositories which are extended to special needs of this thesis.

- `HoloServer`[1]: Python 3 Flask REST server to communicate with HoloClient and utilities to support the labeling process of 2D and 3D data for object detection training.
- `HoloClient`[2]: C# (Visual Studio 2019) Unity 2019.4 implementation of REST client to extract and send 2D and 3D scenes to HoloServer.
- `yolov3-tf2`[3]: Python 3 Tensorflow 2 implementation of YOLOv3 and YOLOv3-tiny. Original training process is extended with data augmentation. A labeling tool is added to simplify the labeling of a complete dataset with YOLOv3.
- `votenet`[4]: Python 3 Pytorch 1.1 implementation of VoteNet. An additional dataset class is added to add support for the newly created AR-2/3 dataset. This implementation automatically returns accuracy measures.
- `review_object_detection_metrics`[5]: Python 3 implementation of 2D object detection accuracy metrics (including MS COCO and Pascal VOC). Used to calculate accuracy metrics for 2D data. Only a minor bug fix is added.

Object Detection Results

During the object detection experiment condition specific validation accuracy is calculate for all runs. Resulting table are to extensive to be shown in Chapter 5, that is why they are available here.

2D Object Detection

To evaluate the difference between the overall and the condition specific accuracy, results are calculated for the best five runs per augmentation level. Then the difference between the overall and the condition specific accuracy is calculated. The results are show in Figure 1, 2, and 3. In comparison higher accuracy values are colored green, lower ones in red. Detailed results for the best run per augmentation level on the test dataset are summarized in Figure 4.

[1] https://git.geo.tuwien.ac.at/markus-kattenbeck/holo2_server, accessed 1-August-2021
[2] https://git.geo.tuwien.ac.at/markus-kattenbeck/holo2_client, accessed 1-August-2021
[3] https://git.geo.tuwien.ac.at/sherrmann/yolov3-tf2, accessed 1-August-2021
[4] https://git.geo.tuwien.ac.at/sherrmann/votenet, accessed 1-August-2021
[5] https://git.geo.tuwien.ac.at/sherrmann/review_object_detection_metrics, accessed 1-August-2021

			mAP				IoU 0.25					IoU 0.5					IoU 0.75				
		run	AP	AP25	AP50	AP75	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table
full aug	Indoor-night	41	0.0373	0.0704	0.0695	0.0334	0.1328	0.0507	0.1179	0.0445	0.0058	0.1328	0.0458	0.1199	0.0455	0.0071	0.0670	0.0357	0.0485	0.0292	-0.0001
		40	0.0245	0.0538	0.0506	0.0122	0.0212	0.1396	0.0396	0.0602	0.0082	0.0212	0.1358	0.0396	0.0597	0.0117	0.0111	0.0211	0.0071	0.0539	-0.0119
		19	0.0049	0.0064	0.0058	0.0149	0.0023	0.0168	-0.0334	-0.0060	0.0526	0.0023	0.0176	-0.0334	-0.0075	0.0562	0.0023	0.0023	-0.0163	0.0390	0.0444
		17	-0.0047	-0.0021	-0.0023	-0.0075	0.0545	-0.0446	-0.0532	0.0236	0.0094	0.0545	-0.0446	-0.0532	0.0231	0.0140	0.0373	-0.0614	-0.0233	0.0391	-0.0264
		21	0.0399	0.0753	0.0739	0.0366	0.0037	0.1632	0.0461	0.1131	0.0506	0.0037	0.1632	0.0461	0.1123	0.0579	0.0013	0.0753	0.0095	0.0899	0.0216
	Indoor-sun	41	-0.0024	-0.0101	-0.0108	0.0073	-0.0456	0.0279	-0.0193	-0.0158	0.0021	-0.0456	0.0308	-0.0199	-0.0168	0.0021	-0.0048	0.0176	0.0312	-0.0330	0.0285
		40	0.0016	-0.0027	-0.0068	0.0106	-0.0120	-0.0004	0.0296	-0.0191	-0.0119	-0.0120	-0.0024	0.0296	-0.0195	-0.0161	-0.0032	0.0046	0.0415	-0.0156	0.0292
		19	0.0037	0.0110	0.0110	0.0014	-0.0044	0.0237	0.0337	0.0262	-0.0245	-0.0044	0.0224	0.0337	0.0248	-0.0222	-0.0044	-0.0002	0.0123	0.0052	-0.0063
		17	0.0010	0.0036	0.0015	0.0007	-0.0176	0.0025	0.0254	0.0179	-0.0101	-0.0176	0.0025	0.0254	0.0174	-0.0106	-0.0121	-0.0193	0.0066	0.0176	-0.0061
		21	0.0125	0.0168	0.0182	0.0226	-0.0006	0.0432	0.0369	-0.0022	0.0068	-0.0006	0.0432	0.0369	0.0013	0.0054	0.0002	0.0236	0.0617	-0.0097	0.0242
	Outdoor-night	41	0.0051	0.0042	0.0048	0.0040	-0.0024	-0.0393	-0.0121	0.0727	0.0022	-0.0024	-0.0363	-0.0128	0.0717	0.0032	-0.0204	-0.0248	-0.0404	0.0728	0.0368
		40	-0.0039	-0.0016	-0.0055	-0.0075	0.0047	-0.0289	-0.0304	0.0202	0.0265	0.0047	-0.0301	-0.0304	0.0197	0.0254	0.0003	-0.0366	-0.0090	0.0088	0.0029
		19	0.0024	0.0048	0.0029	-0.0051	0.0040	-0.0212	0.0105	0.0357	-0.0053	0.0040	-0.0225	-0.0265	0.0342	-0.0076	0.0040	-0.0088	0.0010	0.0024	-0.0055
		17	0.0056	0.0119	0.0083	0.0027	0.0013	-0.0161	0.0368	0.0279	0.0095	0.0013	-0.0161	0.0368	0.0274	0.0074	0.0120	0.0049	0.0120	-0.0156	0.0051
		21	-0.0079	-0.0057	-0.0068	-0.0121	0.0007	-0.0654	-0.0049	0.0407	0.0002	0.0007	-0.0654	-0.0049	0.0374	-0.0026	-0.0009	-0.0708	-0.0086	0.0112	0.0084
	Outdoor-sun	41	-0.0568	-0.0861	-0.0879	-0.0717	-0.1899	-0.0393	-0.0864	-0.1015	-0.0132	-0.1899	-0.0416	-0.0871	-0.1006	-0.0166	-0.0981	-0.0433	-0.0649	-0.0844	-0.0931
		40	-0.0363	-0.0590	-0.0586	-0.0394	-0.0552	-0.1104	-0.0389	-0.0613	-0.0292	-0.0552	-0.1043	-0.0389	-0.0601	-0.0276	-0.0293	-0.0191	-0.0552	-0.0624	-0.0349
		19	-0.0187	-0.0265	-0.0248	-0.0245	-0.0095	-0.0199	-0.0106	-0.0563	-0.0363	-0.0095	-0.0180	-0.0106	-0.0528	-0.0404	-0.0095	-0.0001	-0.0026	-0.0690	-0.0499
		17	-0.0132	-0.0214	-0.0215	-0.0152	-0.0765	0.0582	-0.0089	-0.0693	-0.0104	-0.0765	0.0582	-0.0089	-0.0680	-0.0124	-0.0675	0.0494	-0.0051	-0.0517	0.0022
		21	-0.0540	-0.0898	-0.0891	-0.0587	-0.0205	-0.1411	-0.0782	-0.1520	-0.0570	-0.0205	-0.1411	-0.0782	-0.1519	-0.0598	-0.0047	-0.0352	-0.0828	-0.1062	-0.0719

Figure 1: Difference between overall validation accuracy and conditions specific ones for **full augmentation**.

			mAP				IoU 0.25					IoU 0.5					IoU 0.75				
		run	AP	AP25	AP50	AP75	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table
Medium Aug	Indoor-night	53	0.0272	0.0308	0.0321	0.0436	0.0524	0.0582	0.0093	0.0150	0.0190	0.0524	0.0618	0.0093	0.0145	0.0233	0.0453	0.0580	0.0098	0.0471	0.0577
		50	-0.0004	-0.0067	-0.0030	0.0037	0.0217	-0.0041	-0.0729	0.0136	0.0082	0.0217	-0.0041	-0.0716	0.0129	0.0146	0.0152	-0.0240	-0.0504	0.0359	0.0332
		54	0.0204	0.0368	0.0391	0.0128	0.0469	0.0865	-0.0082	0.0289	0.0302	0.0469	0.0849	-0.0062	0.0282	0.0344	0.0165	0.0278	-0.0440	0.0262	0.0532
		51	0.0228	0.0431	0.0442	0.0137	0.0189	0.1100	0.0071	0.0311	0.0483	0.0189	0.1100	0.0110	0.0305	0.0558	0.0300	0.0093	-0.0295	0.0322	0.0296
		52	0.0036	0.0005	0.0010	0.0128	0.0405	-0.0382	-0.0511	0.0141	0.0372	0.0405	-0.0388	-0.0498	0.0131	0.0468	0.0633	-0.0258	-0.0361	0.0210	0.0382
	Indoor-sun	53	-0.0033	-0.0098	-0.0092	-0.0025	-0.0465	-0.0061	0.0036	0.0050	-0.0050	-0.0465	-0.0072	0.0036	0.0045	-0.0076	-0.0165	-0.0175	0.0307	0.0000	-0.0413
		50	-0.0097	-0.0044	-0.0042	-0.0214	-0.0291	0.0131	0.0129	-0.0207	0.0020	-0.0291	0.0131	0.0124	-0.0214	0.0009	-0.0277	0.0030	-0.0129	-0.0432	-0.0383
		54	-0.0105	-0.0142	-0.0134	-0.0120	-0.0379	-0.0188	0.0046	-0.0096	-0.0094	-0.0379	-0.0211	0.0040	-0.0103	-0.0106	-0.0043	0.0119	-0.0089	-0.0393	-0.0334
		51	-0.0020	-0.0057	-0.0073	0.0003	0.0017	-0.0043	0.0043	-0.0146	-0.0157	0.0017	-0.0043	0.0031	-0.0152	-0.0199	-0.0041	-0.0017	0.0004	-0.0256	0.0228
		52	-0.0105	-0.0125	-0.0156	-0.0091	-0.0642	-0.0211	0.0189	-0.0042	0.0083	-0.0642	-0.0216	0.0185	-0.0052	0.0049	-0.0624	0.0062	0.0271	-0.0105	-0.0034
	Outdoor-night	53	-0.0083	0.0011	0.0005	-0.0215	0.0367	-0.0189	-0.0007	-0.0064	-0.0053	0.0367	-0.0201	-0.0007	-0.0070	-0.0054	-0.0211	-0.0208	-0.0250	-0.0347	-0.0120
		50	0.0119	0.0251	0.0227	0.0070	0.0255	0.0031	0.0643	0.0364	-0.0037	0.0255	0.0031	0.0638	0.0358	-0.0064	0.0167	-0.0097	0.0258	0.0328	-0.0056
		54	-0.0077	-0.0022	-0.0047	-0.0155	0.0412	-0.0345	-0.0111	0.0046	-0.0114	0.0412	-0.0353	-0.0117	0.0039	-0.0112	-0.0136	-0.0495	0.0096	-0.0087	-0.0129
		51	-0.0038	0.0023	0.0014	-0.0026	0.0313	-0.0486	-0.0014	0.0396	-0.0093	0.0313	-0.0486	-0.0026	0.0391	0.0082	-0.0216	-0.0387	0.0057	0.0331	0.0090
		52	0.0061	0.0212	0.0194	-0.0040	0.0520	0.0332	0.0404	0.0151	-0.0345	0.0520	0.0326	0.0399	0.0141	-0.0379	0.0130	-0.0115	0.0129	0.0065	-0.0447
	Outdoor-sun	53	-0.0248	-0.0218	-0.0213	-0.0371	-0.0430	-0.0332	-0.0121	-0.0136	-0.0069	-0.0430	-0.0343	-0.0121	-0.0121	-0.0095	-0.0281	-0.0550	-0.0332	-0.0292	-0.0432
		50	-0.0062	-0.0142	-0.0137	-0.0044	-0.0189	-0.0120	-0.0043	-0.0293	-0.0065	-0.0189	-0.0120	-0.0047	-0.0273	-0.0093	-0.0115	0.0248	0.0180	-0.0349	-0.0011
		54	-0.0125	-0.0220	-0.0192	-0.0065	-0.0506	-0.0331	0.0146	-0.0239	-0.0169	-0.0506	-0.0296	0.0140	-0.0220	-0.0211	-0.0148	-0.0011	0.0238	0.0050	-0.0508
		51	-0.0289	-0.0422	-0.0428	-0.0290	-0.0525	-0.0571	-0.0100	-0.0561	-0.0354	-0.0525	-0.0571	-0.0112	-0.0544	-0.0396	-0.0166	-0.0052	0.0008	-0.0549	-0.0929
		52	0.0036	0.0005	0.0010	0.0128	-0.0298	0.0261	-0.0082	-0.0253	-0.0124	0.0405	-0.0388	-0.0498	0.0131	0.0468	-0.0235	0.0062	-0.0157	-0.0301	-0.0068

Figure 2: Difference between overall validation accuracy and conditions specific ones for **medium augmentation**.

			mAP				IoU 0.25					IoU 0.5					IoU 0.75				
		run	AP	AP25	AP50	AP75	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table
no aug	Indoor-night	11	0.0385	0.0402	0.0432	0.0542	0.0268	0.0650	0.0314	0.0571	0.0206	0.0286	0.0642	0.0330	0.0571	0.0243	0.0565	0.0838	0.0079	0.1030	0.0379
		7	0.0033	0.0083	0.0077	0.0104	0.0129	0.0113	-0.0196	0.0271	0.0099	0.0129	0.0088	-0.0160	0.0294	0.0133	0.0184	0.0258	-0.0330	0.0210	0.0073
		15	0.0462	0.0601	0.0635	0.0670	0.0484	0.0901	0.0257	0.0771	0.0594	0.0477	0.0910	0.0291	0.0771	0.0690	0.0442	0.1015	-0.0065	0.1112	0.0936
		6	0.0011	0.0051	0.0079	0.0066	0.0107	0.0150	-0.0293	0.0136	0.0156	0.0121	0.0146	-0.0284	0.0126	0.0205	0.0226	0.0091	-0.0489	0.0424	0.0213
		9	0.0405	0.0569	0.0607	0.0482	0.0546	0.1079	0.0419	0.0282	0.0518	0.0546	0.1066	0.0433	0.0321	0.0642	0.0297	0.0563	0.0343	0.0734	0.0586
	Indoor-sun	11	-0.0062	-0.0077	-0.0091	-0.0142	-0.0204	0.0064	-0.0086	-0.0157	-0.0005	-0.0209	0.0056	-0.0081	-0.0157	-0.0037	-0.2117	-0.0843	0.0800	0.0880	0.0714
		7	-0.0032	0.0001	-0.0008	-0.0138	-0.0029	0.0110	0.0004	-0.0100	0.0020	-0.0029	0.0085	-0.0009	-0.0116	-0.0003	-0.1373	0.1011	-0.1057	-0.0140	0.0890
		15	-0.0031	-0.0048	-0.0043	-0.0084	-0.0295	0.0096	-0.0200	0.0186	-0.0026	-0.0277	0.0087	-0.0211	0.0186	-0.0080	-0.0094	0.0043	-0.0594	0.0101	0.0180
		6	-0.0084	-0.0068	-0.0042	-0.0204	-0.0627	-0.0236	0.0450	0.0144	-0.0069	-0.0632	-0.0240	0.0463	0.0134	-0.0065	-0.1741	-0.0158	-0.3903	0.4089	0.0926
		9	-0.0002	-0.0074	-0.0093	0.0098	0.0532	-0.0607	-0.0082	-0.0130	-0.0081	0.0532	-0.0583	-0.0087	-0.0153	-0.0127	-0.4047	-0.0473	-0.0359	0.2806	0.2663
	Outdoor-night	11	-0.0152	-0.0095	-0.0110	-0.0182	0.0039	-0.0321	-0.0114	0.0014	-0.0095	0.0034	-0.0330	-0.0125	0.0014	-0.0093	-0.0279	-0.0401	-0.0181	-0.0191	0.0157
		7	0.0039	0.0082	0.0052	0.0023	0.0229	-0.0161	0.0175	0.0143	0.0027	0.0229	-0.0170	0.0163	0.0127	0.0030	-0.0193	-0.0744	0.0266	0.0558	0.0083
		15	-0.0165	-0.0112	-0.0116	-0.0200	0.0148	-0.0591	-0.0071	0.0200	-0.0247	0.0141	-0.0600	-0.0082	0.0200	-0.0270	-0.0091	-0.0716	0.0245	-0.0118	-0.0456
		6	0.0075	0.0072	0.0086	0.0013	0.0573	-0.0050	-0.0050	-0.0101	-0.0012	0.0568	-0.0038	-0.0063	-0.0111	-0.0037	0.0367	-0.0472	-0.0072	-0.0137	-0.0099
		9	-0.0113	-0.0065	-0.0065	-0.0255	-0.0339	-0.0064	0.0267	-0.0005	-0.0186	-0.0339	-0.0080	0.0262	-0.0027	-0.0201	-0.0738	0.0007	0.0265	-0.0303	-0.0458
	Outdoor-sun	11	-0.0284	-0.0227	-0.0234	-0.0484	-0.0104	-0.0393	-0.0114	-0.0429	-0.0095	-0.0109	-0.0369	-0.0125	-0.0429	-0.0124	-0.0401	-0.0430	-0.0032	-0.0864	-0.0718
		7	-0.0142	-0.0166	-0.0164	-0.0100	-0.0329	-0.0061	0.0018	-0.0314	-0.0145	-0.0329	-0.0006	0.0006	-0.0308	-0.0167	-0.0184	0.0025	0.0148	-0.0485	-0.0191
		15	-0.0365	-0.0456	-0.0434	-0.0612	-0.0338	-0.0405	0.0014	-0.1157	-0.0394	-0.0345	-0.0397	0.0003	-0.1157	-0.0421	-0.0562	-0.0463	0.0153	-0.1290	-0.0975
		6	-0.0068	-0.0055	-0.0084	-0.0065	-0.0053	0.0136	-0.0107	-0.0185	-0.0065	-0.0058	0.0132	-0.0120	-0.0161	-0.0089	-0.0079	-0.0344	0.0185	-0.0304	0.0030
		9	-0.0381	-0.0430	-0.0457	-0.0616	-0.0739	-0.0407	-0.0611	-0.0146	-0.0249	-0.0739	-0.0413	-0.0616	-0.0147	-0.0332	-0.0424	-0.0548	-0.0907	-0.0449	-0.0572

Figure 3: Difference between overall validation accuracy and conditions specific ones for **no augmentation**.

			mAP				IoU 0.25					IoU 0.5					IoU 0.75				
			AP	AP25	AP50	AP75	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table
41	full aug	overall	0.3600	0.6202	0.6173	0.3775	0.2244	0.4586	0.8039	0.6457	0.9682	0.2244	0.4529	0.8039	0.6457	0.9678	0.1188	0.1570	0.5855	0.4373	0.6107
53	medium aug		0.5937	0.9411	0.9351	0.6750	0.8418	0.9596	0.9686	0.9496	0.9860	0.8418	0.9593	0.9686	0.9470	0.9852	0.5165	0.6844	0.8482	0.7478	0.6294
11	no aug		0.5953	0.9567	0.9526	0.6528	0.9718	0.9443	0.9603	0.9454	0.9616	0.9698	0.9430	0.9603	0.9448	0.9610	0.5668	0.6642	0.7130	0.7320	0.6293
41	full aug	Indoor-night	0.0416	0.0798	0.0781	0.0451	0.1347	0.0871	0.0868	0.0943	-0.0037	0.1347	0.0814	0.0868	0.0943	-0.0041	0.0775	0.0354	0.0742	0.0757	-0.0390
		Indoor-sun	-0.0018	-0.0018	-0.0013	-0.0006	-0.0327	0.0500	0.0054	-0.0271	-0.0046	-0.0327	0.0497	0.0054	-0.0271	-0.0035	0.0016	0.0324	-0.0193	-0.0299	0.0288
		Outdoor-night	0.0055	0.0055	0.0039	0.0118	-0.0005	-0.0314	0.0082	0.0357	0.0155	-0.0005	-0.0317	0.0082	0.0357	0.0152	-0.0175	-0.0153	0.0198	0.0502	0.0121
		Outdoor-sun	-0.0662	-0.1058	-0.1037	-0.0884	-0.2099	-0.1057	-0.1004	-0.1029	-0.0104	-0.2099	-0.1005	-0.1004	-0.1029	-0.0107	-0.1429	-0.0628	-0.0983	-0.1088	-0.0208
53	medium aug	Indoor-night	0.0290	0.0405	0.0393	0.0405	0.0761	0.0639	0.0086	0.0282	0.0257	0.0761	0.0651	0.0086	0.0297	0.0249	0.0572	0.0358	0.0179	0.0305	0.0735
		Indoor-sun	-0.0047	-0.0120	-0.0112	0.0025	-0.0354	-0.0089	0.0043	-0.0104	-0.0095	-0.0354	-0.0093	0.0043	-0.0105	-0.0103	-0.0465	0.0302	0.0333	-0.0220	-0.0075
		Outdoor-night	-0.0119	-0.0017	-0.0031	-0.0242	0.0204	-0.0289	-0.0014	0.0039	-0.0026	0.0204	-0.0293	-0.0014	0.0041	-0.0010	0.0111	-0.0655	-0.0147	0.0016	-0.0428
		Outdoor-sun	-0.0254	-0.0264	-0.0292	-0.0417	-0.0611	-0.0261	-0.0114	-0.0218	-0.0119	-0.0611	-0.0264	-0.0114	-0.0244	-0.0127	-0.0349	-0.0282	-0.0420	-0.0363	-0.0547
11	no aug	Indoor-night	0.0367	0.0481	0.0481	0.0464	0.0275	0.0729	0.0335	0.0625	0.0442	0.0297	0.0716	0.0335	0.0643	0.0436	0.0269	0.0706	0.0068	0.0712	0.0609
		Indoor-sun	-0.0074	-0.0079	-0.0053	-0.0101	-0.0168	-0.0043	-0.0068	-0.0104	-0.0014	-0.0170	-0.0055	-0.0068	-0.0109	0.0004	-0.0152	-0.0157	-0.0106	-0.0149	0.0086
		Outdoor-night	-0.0195	-0.0169	-0.0155	-0.0314	0.0046	-0.0386	-0.0154	-0.0132	-0.0221	0.0027	-0.0398	-0.0154	-0.0138	-0.0227	-0.0517	-0.0375	-0.0238	0.0078	-0.0427
		Outdoor-sun	-0.0206	-0.0230	-0.0232	-0.0252	-0.0154	-0.0300	-0.0111	-0.0389	-0.0194	-0.0157	-0.0265	-0.0111	-0.0395	-0.0200	0.0134	-0.0239	0.0049	-0.0787	-0.0602

Figure 4: YOLOv3 test accuracy for best model per data augmentation level. Upper part presents the absolute overall accuracy. Higher (green) values are better. The lower three parts show the difference between the overall accuracy and condition specific results. Lower values (green) are better.

3D Object Detection

Similar to the 2D domain also for all 3D runs overall - condition specific accuracy differences are calculated and presented in Figure 5 to 8. Test results for the best run are shown in Figure 9.

		mAP			IoU 0.25					IoU 0.5					IoU 0.75				
	run	AP25	AP50	AP75	chair	cup	monitor	pottedpla	table	chair	cup	monitor	pottedpla	table	chair	cup	monitor	pottedpla	table
Indoor-night	21	0.0059	-0.0196	0.0116	0.0000	0.0103	0.0000	0.0000	0.0193	-0.0026	-0.1058	0.0000	-0.0017	0.0122	0.0304	-0.0787	0.0190	-0.0089	0.0964
	9	-0.0034	-0.0301	0.0005	-0.0004	-0.0223	-0.0005	-0.0007	0.0067	-0.0031	-0.1488	-0.0016	-0.0016	0.0048	0.0056	-0.1245	-0.0071	0.0024	0.1258
	13	-0.0061	-0.0154	0.0216	-0.0024	-0.0248	-0.0088	-0.0007	0.0060	-0.0040	-0.0693	-0.0135	-0.0017	0.0116	0.0393	-0.0195	-0.0197	0.0249	0.0830
	1	0.0036	-0.0257	0.0178	-0.0068	0.0195	-0.0001	0.0000	0.0053	-0.0098	-0.1110	-0.0016	-0.0026	-0.0036	0.0153	-0.0762	0.0056	0.0090	0.1354
	17	-0.0078	-0.0142	0.0257	-0.0011	-0.0407	-0.0007	-0.0007	0.0043	-0.0031	-0.0695	-0.0014	-0.0016	0.0045	0.0370	-0.0601	0.0126	0.0034	0.1355
	19	-0.0077	-0.0230	0.0261	0.0009	-0.0479	0.0021	0.0000	0.0064	-0.0015	-0.1211	0.0014	-0.0007	0.0068	0.0094	-0.0075	-0.0107	0.0093	0.1298
	5	-0.0096	-0.0250	0.0378	0.0000	-0.0552	0.0014	0.0000	0.0057	-0.0009	-0.1233	0.0006	-0.0007	-0.0007	0.0073	-0.0159	0.0584	0.0102	0.1289
	3	0.0059	-0.0156	0.0282	0.0000	0.0238	-0.0008	-0.0007	0.0073	-0.0010	-0.0626	-0.0068	-0.0017	-0.0059	0.0671	-0.0483	0.0226	0.0141	0.0854
	23	-0.0115	-0.0220	0.0381	0.0000	-0.0627	-0.0014	0.0000	0.0064	-0.0008	-0.1128	-0.0030	-0.0007	0.0074	0.0139	-0.0009	-0.0114	0.0117	0.1773
	22	-0.0093	-0.0246	0.0395	0.0000	-0.0529	-0.0029	0.0000	0.0094	-0.0027	-0.1136	-0.0093	-0.0018	0.0042	0.0442	-0.0131	0.0543	0.0077	0.1045
	18	-0.0129	-0.0137	0.0336	-0.0009	-0.0689	-0.0026	-0.0007	0.0088	-0.0041	-0.1011	-0.0076	-0.0025	0.0467	0.0514	-0.0450	-0.0150	0.0190	0.1577
	20	-0.0142	-0.0416	0.0202	0.0000	-0.0790	0.0014	0.0000	0.0064	0.0013	-0.2201	-0.0018	-0.0033	0.0161	-0.0420	-0.0003	0.0032	0.0214	0.1189
	7	-0.0156	-0.0415	0.0418	-0.0001	-0.0823	0.0014	0.0000	0.0028	-0.0022	-0.2116	-0.0008	-0.0025	0.0095	0.0446	-0.0007	0.0026	0.0194	0.1432
	12	-0.0115	-0.0398	0.0221	0.0000	-0.0561	-0.0071	0.0000	0.0057	-0.0019	-0.1805	-0.0087	-0.0178	0.0098	0.0585	-0.0007	0.0115	-0.0393	0.0807
	24	-0.0132	-0.0254	0.0191	0.0000	-0.0684	-0.0029	-0.0007	0.0058	-0.0020	-0.1499	-0.0036	-0.0019	0.0305	0.0153	0.0016	-0.0302	0.0147	0.0943
	6	-0.0141	-0.0252	0.0492	0.0000	-0.0760	0.0000	0.0000	0.0057	-0.0020	-0.1341	-0.0045	-0.0019	0.0164	0.0610	-0.0001	0.0090	0.0056	0.1703
	11	-0.0139	-0.0401	-0.0007	0.0000	-0.0775	0.0007	0.0000	0.0071	0.0000	-0.1664	-0.0030	-0.0312	0.0001	0.0428	-0.0323	-0.0062	-0.1091	0.1013
	14	-0.0074	-0.0232	0.0095	0.0015	-0.0424	-0.0015	-0.0007	0.0060	0.0002	-0.1196	-0.0051	-0.0024	0.0111	0.1005	-0.1076	-0.0786	0.0242	0.1092
	16	-0.0147	-0.0408	-0.0117	0.0003	-0.0718	-0.0064	-0.0007	0.0050	0.0065	-0.1639	-0.0064	-0.0726	0.0322	-0.0516	-0.0074	-0.0490	-0.0105	0.0603
	10	0.0312	-0.0216	0.0290	0.0029	0.1393	-0.0087	-0.0007	0.0233	-0.0025	-0.0970	-0.0182	-0.0015	0.0113	0.0184	-0.0185	0.0052	-0.0037	0.1435
	15	0.0771	-0.0356	0.0216	0.0000	0.0343	-0.0064	0.3387	0.0189	-0.0019	-0.1677	-0.0080	-0.0142	0.0136	0.0415	-0.0078	-0.0400	0.0096	0.1048
	2	-0.0096	-0.0312	0.0467	-0.0002	-0.0516	-0.0007	-0.0002	0.0047	-0.0014	-0.1565	-0.0007	-0.0022	0.0049	0.0047	-0.0083	0.0138	0.0083	0.2148
	4	-0.0094	-0.0227	0.0313	0.0057	-0.0708	-0.0069	0.0026	0.0224	0.0023	-0.2004	-0.0298	-0.0012	0.1157	0.0714	0.0000	-0.0733	0.1147	0.0438
	8	-0.0289	0.0120	0.0205	0.0001	-0.1429	-0.0072	-0.0001	0.0056	-0.0030	-0.0416	0.0011	-0.0053	0.1090	0.0382	0.0000	-0.0100	0.0667	0.0077

Figure 5: Difference between overall validation accuracy and indoor-night condition.

		mAP			IoU 0.25					IoU 0.5					IoU 0.75				
	run	AP25	AP50	AP75	chair	cup	monitor	pottedpla	table	chair	cup	monitor	pottedpla	table	chair	cup	monitor	pottedpla	table
Indoor-sun	21	-0.0028	-0.0136	-0.0353	0.0000	-0.0139	0.0000	0.0000	0.0001	-0.0025	-0.0725	0.0000	-0.0017	0.0086	-0.0301	-0.0567	-0.0523	0.0039	-0.0415
	9	-0.0045	-0.0155	-0.0236	0.0033	-0.0280	-0.0005	-0.0007	0.0035	0.0006	-0.0875	-0.0016	-0.0016	0.0125	-0.0153	-0.0749	-0.0249	-0.0164	0.0137
	13	0.0027	-0.0078	-0.0534	0.0078	-0.0256	0.0255	-0.0007	0.0065	0.0061	-0.0899	0.0139	-0.0017	0.0324	-0.0584	-0.1133	-0.0294	0.0544	-0.1204
	1	0.0016	-0.0066	-0.0635	0.0249	-0.0157	0.0000	0.0000	-0.0010	0.0219	-0.0686	-0.0015	-0.0026	0.0176	-0.0201	-0.1543	-0.0663	-0.0252	-0.0515
	17	-0.0074	-0.0229	-0.0439	0.0040	-0.0412	0.0021	-0.0007	-0.0010	0.0020	-0.1278	0.0014	-0.0016	0.0115	-0.0424	-0.0942	-0.0184	0.0082	-0.0727
	19	-0.0043	-0.0345	-0.0557	-0.0004	-0.0213	0.0021	0.0000	-0.0021	-0.0027	-0.1817	0.0014	-0.0007	0.0113	-0.0659	-0.0301	-0.1104	-0.0188	-0.0534
	5	-0.0057	-0.0358	-0.0263	0.0000	-0.0344	0.0014	0.0000	0.0042	-0.0009	-0.1979	0.0006	-0.0007	0.0199	-0.0542	-0.0184	-0.0832	0.0012	0.0231
	3	-0.0111	-0.0294	-0.0424	0.0000	-0.0578	0.0000	-0.0007	0.0033	0.0001	-0.1628	-0.0100	-0.0017	0.0277	-0.0536	-0.0597	-0.0823	-0.0043	-0.0121
	23	-0.0110	-0.0403	-0.0273	0.0000	-0.0541	0.0014	0.0000	-0.0021	-0.0008	-0.2047	-0.0001	-0.0007	0.0046	-0.0458	-0.0332	-0.0642	-0.0064	0.0131
	22	-0.0042	-0.0238	-0.0654	0.0000	-0.0186	-0.0033	0.0000	0.0011	-0.0026	-0.1105	-0.0098	-0.0018	0.0055	-0.0703	-0.0014	-0.1064	-0.0178	-0.1310
	18	-0.0109	-0.0217	-0.0354	0.0047	-0.0608	0.0003	-0.0007	0.0023	0.0015	-0.1136	-0.0088	-0.0025	0.0149	-0.0151	-0.0055	-0.0021	-0.0156	-0.1385
	20	-0.0097	-0.0298	-0.0395	0.0000	-0.0479	0.0014	0.0000	-0.0021	-0.0018	-0.1547	-0.0018	-0.0033	0.0127	-0.0423	-0.0010	-0.0826	-0.0030	-0.0684
	7	-0.0094	-0.0376	-0.0594	0.0001	-0.0486	0.0015	0.0000	0.0001	-0.0021	-0.1910	-0.0008	-0.0025	0.0083	-0.0709	-0.0012	-0.1174	-0.0087	-0.0986
	12	-0.0019	-0.0345	-0.0290	0.0000	-0.0339	0.0243	0.0000	0.0003	-0.0019	-0.2259	0.0228	0.0274	0.0052	-0.0772	-0.0094	-0.1115	0.0304	0.0228
	24	-0.0079	-0.0358	-0.0185	0.0000	-0.0416	0.0029	-0.0007	0.0000	-0.0020	-0.1864	0.0051	-0.0019	0.0060	-0.0821	-0.0036	-0.0561	-0.0132	0.0627
	6	-0.0096	-0.0411	-0.0136	0.0000	-0.0529	0.0000	0.0000	0.0051	-0.0020	-0.1995	-0.0045	-0.0019	0.0027	-0.0367	0.0004	-0.0349	-0.0014	0.0044
	11	0.0290	-0.0223	-0.0594	0.0000	0.0574	0.0007	0.0509	0.0360	0.0000	-0.1542	-0.0030	0.0196	0.0259	-0.0655	-0.0467	-0.1107	-0.0546	-0.0197
	14	-0.0065	-0.0179	-0.0307	-0.0001	-0.0451	0.0102	-0.0007	0.0034	-0.0015	-0.1265	0.0234	-0.0024	0.0172	-0.0578	-0.0424	0.0034	0.0064	-0.0629
	16	-0.0056	-0.0316	-0.0296	0.0012	-0.0527	0.0250	-0.0007	-0.0007	-0.0046	-0.1852	0.0250	0.0049	0.0019	-0.0759	-0.0068	-0.0217	0.0094	-0.0530
	10	-0.0035	-0.0168	-0.0436	0.0000	-0.0376	0.0228	-0.0007	-0.0021	-0.0053	-0.1012	0.0132	-0.0015	0.0110	-0.0393	-0.0419	-0.0252	0.0030	-0.1148
	15	0.0056	-0.0146	-0.0306	0.0000	-0.0273	0.0250	0.0281	0.0021	-0.0018	-0.1310	0.0235	0.0222	0.0144	-0.0238	-0.0254	-0.0771	-0.0058	-0.0211
	2	-0.0168	-0.0222	-0.0606	0.0005	-0.0865	-0.0007	-0.0001	0.0029	-0.0007	-0.1240	-0.0007	-0.0021	0.0165	-0.0652	-0.0359	-0.0792	0.0006	-0.1232
	4	-0.0141	-0.0227	-0.0481	0.0001	-0.0811	0.0106	-0.0014	0.0014	-0.0033	-0.1438	-0.0209	-0.0053	0.0595	-0.1180	-0.0004	-0.0792	0.0316	-0.0747
	8	-0.0230	0.0208	-0.0177	0.0000	-0.1581	0.0271	0.0003	0.0155	-0.0031	-0.0105	0.0253	-0.0021	0.0945	-0.0812	0.0000	-0.0514	0.0439	0.0002

Figure 6: Difference between overall validation accuracy and indoor-sun condition.

		mAP			IoU 0.25					IoU 0.5					IoU 0.75				
	run	AP25	AP50	AP75	chair	cup	monitor	pottedpla	table	chair	cup	monitor	pottedpla	table	chair	cup	monitor	pottedpla	table
Outdoor-night	21	0.0052	0.0221	0.0354	0.0000	0.0262	0.0000	0.0029	-0.0029	0.0026	0.1126	0.0000	0.0052	-0.0101	0.0225	0.0866	0.1185	0.0152	-0.0658
	9	0.0029	0.0178	0.0098	-0.0004	0.0169	-0.0005	0.0021	-0.0036	0.0034	0.0929	-0.0016	0.0052	-0.0106	-0.0251	0.1167	0.0257	0.0109	-0.0792
	13	-0.0009	0.0268	-0.0095	-0.0024	0.0068	-0.0083	0.0022	-0.0025	-0.0004	0.1249	0.0150	0.0053	-0.0111	-0.0189	0.0276	0.0722	-0.0447	-0.0835
	1	-0.0019	0.0142	0.0174	-0.0159	0.0106	0.0000	0.0000	-0.0039	-0.0149	0.0859	0.0018	0.0047	-0.0066	0.0075	0.0996	0.0304	0.0181	-0.0687
	17	0.0105	0.0303	0.0151	-0.0011	0.0537	-0.0007	0.0021	-0.0016	0.0013	0.1505	-0.0014	0.0092	-0.0081	-0.0070	0.0494	0.0569	0.0181	-0.0420
	19	0.0042	0.0309	0.0013	-0.0004	0.0241	-0.0007	0.0000	-0.0021	0.0027	0.1607	-0.0015	0.0021	-0.0096	0.0254	0.0081	0.0394	0.0016	-0.0680
	5	0.0023	0.0293	0.0058	0.0000	0.0160	-0.0014	0.0000	-0.0029	0.0047	0.1537	-0.0022	-0.0007	-0.0092	0.0144	0.0141	0.0666	-0.0047	-0.0614
	3	0.0028	0.0219	-0.0032	0.0000	0.0177	-0.0008	0.0021	-0.0051	0.0015	0.1259	-0.0036	0.0059	-0.0204	0.0443	0.0516	-0.0002	0.0233	-0.1349
	23	-0.0172	0.0310	0.0181	0.0928	-0.0826	-0.0014	0.0000	0.0164	-0.0008	0.1698	-0.0030	0.0021	-0.0131	0.0360	0.0174	0.1158	-0.0003	-0.0785
	22	0.0070	0.0194	-0.0015	0.0000	0.0299	0.0073	0.0000	-0.0022	0.0067	0.0989	0.0009	0.0054	-0.0147	0.0171	0.0199	0.0157	-0.0129	-0.0475
	18	0.0144	0.0347	0.0216	-0.0009	0.0771	-0.0016	0.0021	-0.0047	0.0050	0.1819	-0.0051	0.0082	-0.0165	0.0158	0.0431	0.0367	0.0098	0.0027
	20	0.0026	0.0378	0.0037	0.0000	0.0167	-0.0015	0.0000	-0.0021	0.0030	0.1784	-0.0017	0.0095	-0.0003	0.0163	0.0003	-0.0101	0.0061	0.0061
	7	0.0037	0.0428	0.0113	-0.0001	0.0230	-0.0014	0.0000	-0.0029	0.0028	0.2089	0.0119	0.0041	-0.0136	-0.0071	0.0003	0.0526	0.0058	0.0050
	12	0.0070	0.0287	-0.0044	0.0000	0.0450	-0.0071	0.0001	-0.0029	0.0030	0.1447	0.0033	0.0010	-0.0087	-0.0094	0.0022	0.0056	0.0537	-0.0740
	24	0.0055	0.0362	0.0212	0.0000	0.0253	0.0029	0.0021	-0.0029	0.0022	0.1869	0.0052	0.0066	-0.0201	0.0022	0.0013	0.0757	0.0201	0.0068
	6	0.0015	0.0453	-0.0088	0.0000	0.0046	0.0028	0.0029	-0.0029	0.0030	0.2307	0.0073	0.0058	-0.0206	0.0201	0.0018	0.0286	0.0155	-0.1099
	11	-0.0006	0.0197	0.0138	0.0000	0.0005	-0.0021	0.0000	-0.0014	0.0030	0.1190	-0.0058	-0.0063	-0.0115	0.0297	0.0135	0.0214	0.0381	-0.0338
	14	0.0072	0.0277	0.0204	-0.0004	0.0422	-0.0044	0.0021	-0.0036	0.0066	0.1408	-0.0050	0.0077	-0.0116	0.0348	0.0318	-0.0074	0.0196	0.0233
	16	0.0184	0.0502	0.0121	-0.0001	0.1029	-0.0093	0.0021	-0.0036	-0.0009	0.1751	-0.0033	0.1061	-0.0260	0.0360	0.0053	0.0360	0.0265	-0.0435
	10	0.0015	0.0321	0.0198	0.0000	0.0160	-0.0086	0.0021	-0.0022	0.0038	0.1770	-0.0111	0.0048	-0.0142	0.0312	0.0180	0.0259	0.0407	-0.0168
	15	0.0089	0.0694	0.0226	0.0000	0.0632	-0.0093	-0.0057	-0.0036	0.0089	0.2802	-0.0045	0.0681	-0.0057	-0.0274	0.0092	0.1057	0.0192	0.0062
	2	0.0044	0.0306	0.0230	-0.0002	0.0270	-0.0007	-0.0001	-0.0039	0.0035	0.1445	0.0127	0.0055	-0.0132	-0.0028	0.0305	0.1181	0.0131	-0.0440
	4	0.0189	0.0232	0.0076	0.0000	0.1136	-0.0132	0.0014	-0.0074	0.0061	0.1682	0.0398	0.0084	-0.1063	0.0344	0.0002	0.0012	0.0408	-0.0385
	8	0.0337	-0.0020	0.0024	-0.0001	0.1771	0.0021	0.0028	-0.0133	0.0070	0.0442	-0.0025	0.0144	-0.0728	0.0232	0.0000	0.0215	-0.0307	-0.0017

Figure 7: Difference between overall validation accuracy and outdoor-night condition.

		mAP			IoU 0.25					IoU 0.5					IoU 0.75				
	run	AP25	AP50	AP75	chair	cup	monitor	pottedpla	table	chair	cup	monitor	pottedpla	table	chair	cup	monitor	pottedpla	table
Outdoor-sun	21	0.0077	0.0135	-0.0048	0.0000	0.0412	0.0000	0.0000	-0.0029	0.0028	0.0765	0.0000	-0.0017	-0.0101	0.0003	0.0744	-0.0229	-0.0117	-0.0641
	9	0.0107	0.0139	-0.0026	-0.0004	0.0587	-0.0004	-0.0007	-0.0036	0.0000	0.0788	0.0029	-0.0016	-0.0106	-0.0439	0.1128	0.0233	-0.0059	-0.0993
	13	0.0070	0.0113	0.0129	-0.0024	0.0494	-0.0088	-0.0007	-0.0025	-0.0040	0.0817	-0.0083	-0.0017	-0.0111	0.0266	0.0456	0.0396	-0.0425	-0.0046
	1	0.0023	0.0112	0.0037	-0.0159	0.0311	-0.0001	0.0000	-0.0034	-0.0139	0.0827	-0.0016	-0.0025	-0.0086	-0.0097	0.1062	-0.0161	-0.0003	-0.0616
	17	0.0079	0.0225	-0.0113	-0.0011	0.0436	-0.0007	-0.0007	-0.0016	0.0017	0.1220	-0.0014	-0.0016	-0.0081	-0.0221	0.0822	-0.0170	-0.0176	-0.0822
	19	0.0110	0.0279	-0.0156	-0.0004	0.0580	-0.0007	0.0000	-0.0021	0.0005	0.1407	0.0018	-0.0007	-0.0030	0.0366	0.0085	-0.0294	0.0255	-0.1196
	5	0.0163	0.0254	0.0053	0.0000	0.0857	-0.0014	0.0000	-0.0029	0.0027	0.1326	-0.0022	-0.0007	-0.0055	0.0007	0.0162	0.0074	0.0331	-0.0309
	3	0.0092	0.0119	0.0098	0.0000	0.0522	-0.0006	-0.0007	-0.0051	0.0008	0.0826	-0.0031	-0.0017	-0.0189	-0.0573	0.0457	0.0213	-0.0213	0.0604
	23	0.0172	0.0271	-0.0277	0.0000	0.0897	-0.0014	0.0000	-0.0021	0.0023	0.1472	-0.0030	-0.0007	-0.0103	0.0037	0.0131	-0.0233	-0.0031	-0.1287
	22	0.0047	0.0020	-0.0010	0.0003	0.0258	-0.0005	0.0000	-0.0022	0.0036	0.0296	-0.0069	-0.0018	-0.0147	0.0038	0.0010	0.0059	-0.0377	0.0218
	18	0.0107	0.0202	-0.0105	-0.0009	0.0614	-0.0016	-0.0007	-0.0048	0.0014	0.1082	0.0074	-0.0024	-0.0136	-0.0349	0.0334	-0.0453	-0.0274	0.0218
	20	0.0177	0.0234	-0.0159	0.0000	0.0923	-0.0015	0.0000	-0.0021	0.0013	0.1310	-0.0047	-0.0033	-0.0072	0.0350	-0.0002	-0.0238	-0.0089	-0.0816
	7	0.0166	0.0192	-0.0470	-0.0001	0.0874	-0.0014	0.0000	-0.0029	0.0010	0.1090	-0.0006	-0.0025	-0.0107	-0.0424	-0.0012	-0.0288	-0.0426	-0.1197
	12	0.0121	0.0195	-0.0260	0.0000	0.0734	-0.0100	0.0000	-0.0029	0.0013	0.1296	-0.0115	-0.0130	-0.0088	0.0183	0.0014	-0.0185	-0.0508	-0.0805
	24	0.0106	0.0140	-0.0440	0.0000	0.0592	-0.0029	-0.0007	-0.0029	-0.0020	0.0941	-0.0036	-0.0019	-0.0167	-0.0137	0.0005	-0.0397	-0.0212	-0.1461
	6	0.0187	0.0258	-0.0217	0.0000	0.0978	-0.0014	0.0000	-0.0029	0.0011	0.1599	-0.0058	-0.0019	-0.0244	-0.0098	-0.0013	-0.0047	0.0181	-0.1108
	11	0.0058	0.0227	-0.0275	0.0023	0.0276	-0.0021	0.0029	-0.0014	0.0067	0.1445	-0.0030	-0.0232	-0.0115	0.0486	0.0145	0.0293	-0.1207	-0.1091
	14	0.0076	0.0094	-0.0085	-0.0002	0.0440	-0.0015	-0.0007	-0.0036	0.0014	0.0613	-0.0015	-0.0024	-0.0116	-0.0108	0.0280	-0.0255	-0.0076	-0.0267
	16	0.0074	0.0140	-0.0080	-0.0001	0.0507	-0.0093	-0.0007	-0.0036	-0.0022	0.1695	-0.0093	-0.0614	-0.0265	0.0309	0.0029	0.0326	-0.0632	-0.0430
	10	0.0098	0.0259	-0.0173	0.0000	0.0525	-0.0028	-0.0007	0.0000	0.0000	0.1457	-0.0028	-0.0015	-0.0120	-0.0199	0.0247	-0.0047	-0.0472	-0.0397
	15	-0.0006	0.0062	-0.0298	0.0000	0.0214	-0.0093	-0.0114	-0.0036	0.0014	0.1195	-0.0108	-0.0702	-0.0089	-0.0385	0.0071	0.0082	-0.0110	-0.1148
	2	0.0129	0.0188	-0.0066	-0.0002	0.0697	-0.0007	-0.0002	-0.0039	-0.0014	0.1116	-0.0007	-0.0022	-0.0132	-0.0004	0.0259	0.0331	-0.0157	-0.0757
	4	0.0076	0.0105	-0.0132	0.0000	0.0421	0.0004	-0.0014	-0.0029	-0.0034	0.1315	0.0183	-0.0022	-0.0918	0.0023	0.0001	-0.0026	-0.0974	0.0317
	8	0.0104	-0.0207	-0.0424	-0.0001	0.0762	-0.0129	-0.0001	-0.0111	0.0010	0.0473	-0.0427	-0.0053	-0.1039	-0.0076	-0.0001	-0.0444	-0.1595	-0.0003

Figure 8: Difference between overall validation accuracy and outdoor-sun condition.

		mAP			IoU 0.25					IoU 0.5					IoU 0.75				
	condition	AP25	AP50	AP75	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table	chair	cup	monitor	pottedplant	table
21	overall	0.9884	0.9639	0.7059	1.0000	0.9487	0.9971	1.0000	0.9962	1.0000	0.8327	0.9964	0.9982	0.9920	0.8634	0.1593	0.7932	0.9580	0.7556
	Indoor-night	-0.0089	-0.0161	0.0068	0.0000	-0.0464	0.0000	0.0000	0.0020	0.0000	-0.0758	-0.0007	-0.0018	-0.0022	0.0343	-0.0638	-0.0505	0.0018	0.1119
	Indoor-sun	-0.0057	-0.0137	-0.0484	0.0000	-0.0276	-0.0029	0.0000	0.0020	0.0000	-0.0777	-0.0007	-0.0018	0.0116	-0.0506	-0.1165	-0.0647	0.0111	-0.0214
	Outdoor-night	0.0074	0.0256	0.0106	0.0028	0.0403	-0.0029	0.0006	-0.0038	0.0028	0.1271	-0.0002	0.0031	-0.0049	0.0121	0.0864	0.0409	-0.0042	-0.0820
	Outdoor-sun	0.0079	0.0214	-0.0023	0.0000	0.0433	0.0000	0.0000	-0.0038	0.0000	0.1178	-0.0007	-0.0018	-0.0080	0.0094	0.0875	0.0037	-0.0175	-0.0948

Figure 9: VoteNet test accuracy for best model per data augmentation level. Upper part presents the absolute overall accuracy. Higher (green) values are better. The lower three parts show the difference between the overall accuracy and condition specific results. Lower values (green) are better.