Tampere University

Tomi Koskinen

# ORIENTED BOUNDING BOX DETECTION FOR BIRD'S EYE VIEW CAMERA

Master of Science Thesis
Faculty of Information Technology and Communication Sciences
Examiners: Prof. Esa Rahtu
Dr.Eng. Heikki Huttunen
February 2024

# ABSTRACT

Tomi Koskinen: Oriented Bounding Box Detection for Bird's Eye View Camera
Master of Science Thesis
Tampere University
Master's Degree Programme in Information Technology
February 2024

---

Oriented bounding box detection has seen a lot of development and new methods in the recent years. Almost without exception, the detection target for these methods has been objects in aerial images, taken with drones or satellites. Generally object detection methods use axis-aligned horizontal bounding boxes to localize objects in an image, but object detection from aerial images benefits from using oriented bounding boxes, because they are able to surround the objects with less background clutter and less overlapping with other objects.

This thesis considers another similar perspective, that could also benefit from the use of oriented bounding boxes. This perspective, defined as bird's eye view, can be achieved by using high up surveillance cameras pointed downwards towards the ground. Places like parking lots, container ports and other locations that can have densely packed objects like cars, containers, boats and trucks, can use object detection combined with software to better manage these areas and make operations faster and more efficient. A simple example would be to detect the number of vehicles in different areas of a huge parking lot, and then directing incoming traffic based on that information.

The main objective in this thesis were to find out answers for these research questions: Does the bird's eye view object detection benefit from using oriented bounding boxes? Which open-source oriented bounding box detection method performs the best on bird's eye view images? Are there some oriented bounding box detection method key attributes that are always better than the others? To answer these questions a small dataset of 820 images from the defined perspective was created. Six open-source implementations of oriented bounding box detection methods were trained and evaluated on this created dataset.

Rotated RetinaNet was evaluated to being slightly(1.9 percentage-points) more accurate while using oriented bounding boxes instead of horizontal bounding boxes. The fastest and most accurate method on the created dataset was YOLOv5+Circular Smooth Labeling(CSL), with a 0.405 mean average precision and 112.4 frames per second. This makes YOLOv5+CSL easy to recommend. The key attributes were: angle-free/based, anchor-free/based and single/two-stage. No correlation could be found between detectors' key attributes and their performance on the created dataset.

Keywords: Oriented bounding box, oriented object detection, object detection, rotated bounding box, bird's eye view, computer vision

# TIIVISTELMÄ

Tomi Koskinen: Kiertyneen kohteen tunnistaminen lintuperspektiivi kameralla
Diplomityö
Tampereen yliopisto
Tietotekniikan DI-ohjelma
Helmikuu 2024

---

Kiertyneen kohteen tunnistus on nähnyt paljon kehitystä ja uusia menetelmiä viime vuosina. Lähes poikkeuksetta näiden menetelmien tunnistuskohteena ovat olleet ilmakuvissa olevat kohteet, joiden kuvat on otettu drooneilla tai satelliiteilla. Yleensä kohteentunnistusmenetelmät käyttävät koordinaattiakselien suuntaisia vaakatasossa olevia rajauslaatikoita kohteiden paikallistamiseen kuvassa, mutta kohteentunnistus ilmakuvista hyötyy kiertyneiden rajauslaatikoiden käytöstä, koska ne pystyvät ympäröimään kohteet niin, että rajauslaatikoissa on vähemmän turhaa taustaa ja päällekkäisyyksiä muiden kohteiden kanssa.

Tässä opinnäytetyössä tarkastellaan toista samankaltaista näkökulmaa, joka voisi hyötyä myös kiertyneiden rajauslaatikoiden käytöstä. Tämä lintuperspektiiviksi määritelty näkökulma voidaan saavuttaa käyttämällä korkealle ylös asetettuja, alaspäin maata kohti suunnattuja valvontakameroita. Sellaiset alueet, kuten parkkipaikat, konttisatamat ja muut paikat, joissa voi olla tiheästi asetettuja kohteita, kuten autoja, kontteja, veneitä ja kuorma-autoja, voivat käyttää kohteentunnistusta yhdistettynä tietokoneohjelmistojen kanssa näiden alueiden hallinnan parantamiseksi ja toiminnan nopeuttamiseksi ja tehostamiseksi. Yksinkertainen esimerkki tästä olisi tunnistaa ajoneuvojen määrää valtavan pysäköintialueen eri alueilla ja ohjata sitten saapuvaa liikennettä näiden tietojen perusteella.

Tämän opinnäytetyön päätavoitteena oli löytää vastauksia näihin tutkimuskysymyksiin: Hyötyykö lintuperspektiivistä tehtävä kohteentunnistus kiertyneistä rajauslaatikoista? Mikä avoimen lähdekoodin kiertyneen kohteentunnistuksen menetelmä suoriutuu parhaiten lintuperspektiivikuvilla? Onko olemassa joitakin kiertyneen kohteentunnistusmenetelmien -avainattribuutteja, jotka ovat aina parempia kuin muut? Vastatakseen näihin kysymyksiin, pieni 820 kuvan tietoaineisto luotiin määritellystä perspektiivistä. Tämän luodun tietoaineiston pohjalta opetettiin ja arvioitiin kuusi avoimen lähdekoodin toteutusta kiertyneiden rajauslaatikoiden tunnistusmenetelmistä.

Rotated RetinaNet arvioitiin olevan hieman(1.9 prosenttiyksikköä) tarkempi käyttäessään kiertyneitä rajauslaatikoita vaakatasoisten rajauslaatikoiden sijaan. Luodun tietoaineiston nopein ja tarkin menetelmä oli YOLOv5+Circular Smooth Labeling(CSL), jonka keskimääräinen tarkkuus oli 0,405 ja tunnistusnopeus 112,4 kuvaa sekunnissa. Tämä tekee YOLOv5+CSL:stä helpon suositella. Tärkeimmät avainattribuutit olivat: kulmavapaa tai kulmapohjainen, ankkuriton tai ankkuripohjainen ja yksi- tai kaksivaiheinen. tunnistusmenetelmien avainattribuuttien ja niiden suorituskyvyn välillä ei löytynyt korrelaatiota luodussa tietoaineistossa.

Avainsanat: Kiertynyt rajauslaatikko, kiertynyt kohteentunnistus, kohteentunnistus, pyöritetty rajauslaatikko, lintuperspektiivi, konenäkö

# PREFACE

I am thrilled to present my thesis, which was done in collaboration with Visy Oy. My biggest thanks to Visy and especially Heikki Huttunen for coming up with the topic and providing me with the opportunity to conduct the thesis, and thus allowing me to finish my studies at Tampere University. Thanks also to my thesis supervisor from the university Prof. Esa Rahtu.

Finally, I am very grateful to my friends and family for supporting me throughout my time at Tampere University.

Tampere, 28th February 2024

Tomi Koskinen

# CONTENTS

# LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| AP | Average Precision |
| CE | Cross Entropy |
| CNN | Convolutional neural network |
| CSL | Circular smooth label |
| CSPNet | Cross Stage Partial Network |
| CVAT | Computer Vision Annotation Tool |
| DFL | Distribution Focal Loss |
| DOTA | Large-scale Dataset for Object Detection in Aerial Images |
| ET-head | Efficient Task-aligned Head |
| FL | Focal Loss |
| FN | False Negative |
| FP | False Positive |
| fp16 | 16 bit floating-point |
| FPN | Feature Pyramid Network |
| FPS | Frames per second |
| GBB | Gaussian bounding box |
| GWD | Gaussian Wasserstein distance |
| HBB | Horizontal bounding box |
| IoU | Intersection over union |
| KFIoU | Kalman Filter IoU |
| KLD | Kullback-Leibler Divergence |
| le135 | $135^\circ$ long edge bounding box definition |
| le90 | $90^\circ$ long edge bounding box definition |
| mAP | Mean average precision |
| MS | Multi-scale |
| NMS | Non-maximum suppression |
| OBB | Oriented bounding box |

| | |
|---|---|
| oc | OpenCV bounding box definition |
| OCR | Optical character recognition |
| PAN | Path Aggregation Network |
| PR Curve | Precision-Recall Curve |
| Precision | Object detection performance metric |
| ProbIoU | Probabilistic IoU |
| QBB | Quadrilateral bounding box |
| R-CNN | Region-Based Convolutional Neural Network |
| Recall | Object detection performance metric |
| RoI | Region of interest |
| RPN | Region Proposal Network |
| RPS-RoI-Align | Rotated Position Sensitive RoI Align |
| RR | Random rotation |
| RRoI | Rotated region of interest |
| RSI | Remote sensing image |
| SPP | Spatial Pyramid Pooling |
| SPPF | Spatial Pyramid Pooling Fusion |
| SSD | Single Shot Detector |
| SVM | Support Vector Machine |
| TAL | Task alignment learning |
| TP | True Positive |
| TPL | Tampere Parking Lot |
| YOLO | You Only Look Once |

# 1. INTRODUCTION

Computer vision is a field of artificial intelligence that uses computers to collect interesting information from visual data like digital images, videos or three-dimensional point clouds. The best known task of computer vision is called object detection, which is used to find objects from images or video frames, draw a horizontal bounding box around them and classify them with a label. Nowadays object detection methods are trained to detect objects by feeding images that have labeled/annotated objects in them to a neural network, which moves the network's weight parameters and eventually, with enough training images, the network can hopefully detect those objects from unseen images.

Camera-based object detection has many important and interesting application areas, like detecting vehicles, pedestrians and road signs from a camera on a self-driving vehicle. But sometimes the camera feed is too complicated for generic object detection methods to perform at a required level. One of these situations is aerial images, where the objects that require detecting, like boats or cars, are in such orientations and so densely presented in the image, that the axis-aligned horizontal bounding box has lots of meaningless background or even other objects inside of it. This of course negatively affects the training and detection performance of the network. The solution for this has been oriented object detection, which generally adds an angle parameter, to create an oriented bounding box (OBB). Now this OBB can be rotated to better fit the objects in an aerial image, and remove useless clutter and overlap, thus increasing accuracy.

This thesis considers the high up bird's eye view camera perspective, which can be achieved by placing surveillance cameras in parking lots, container ports and other locations that can have densely packed objects like cars, containers, boats and trucks. The detection of these objects combined with software can be used to better manage these areas and make operations faster and more efficient. A simple example would be to detect the number of vehicles in different areas of a huge parking lot, and then directing incoming traffic based on that information.

The bird's eye view images much more resemble the aerial satellite images used in the popular oriented bounding box dataset DOTA [72] compared to the natural scene images of popular object detection datasets like Microsoft's COCO [39], which suggests that the use of oriented bounding boxes would be more well suited for this use case. The first

research question for this thesis is: **Does the bird's eye view object detection benefit from using oriented bounding boxes?** To answer this question this thesis aims to create a small dataset, featuring bird's eye view images of a parking lot and an annotated vehicle class. Using this dataset, an object detection method will be trained and evaluated using horizontal bounding boxes and oriented bounding boxes.

Lately there has been an increase in the development of oriented object detection methods. Sometimes an open-source implementation is also available for these methods. These developed methods do not consider or evaluate their performance on the defined bird's eye view circumstance, which is why in this thesis, the second research question is: **Which open-source oriented bounding box detection method performs the best on bird's eye view images?** This question is answered by training and evaluating six different open-source OBB detection methods on the created dataset.

Often the methods share some key techniques or attributes that separates the methods into different categories, which leads to the third research question: **Are there some OBB detection method key attributes that are always better than the others?** This is answered by categorizing the used methods based on their key attributes and analyzing their evaluation results on the created dataset.

This thesis is structured as follows: Chapter 2 discusses the basics of object detection and oriented object detection. Chapter 3 introduces all the oriented bounding box detection methods and covers their key attributes. Chapter 4 introduces the dataset and how it was created, and describes the training and evaluation process and their results with the used methods. Then Chapter 5 summarizes and concludes the thesis.

# 2. BACKGROUND

In this chapter the core concepts of modern day object detection are explained and the most influential convolutional neural network object detection methods are introduced. Following this, oriented object detection and the most important topics around it are presented, which provides context for the rest of the thesis.

## 2.1 Object detection

Localizing and classifying objects of interest from a digital image is referred to as object detection. The localization part is usually done by finding the bounding box coordinates, which the object in the image resides inside of. The classifying task's job is to label what is the class of the object inside the bounding box. For example if it is a dog, a cat or a car. Some researchers use the term object localization when the image has only one object of interest to be pinpointed and classified, and object detection when there are multiple objects in the image [48].

Object detection is one of the best known computer vision tasks and as such the development of better and better object detection techniques has been rapid from the first popular neural network implementations to this day. Object detection has a large amount of possible use cases in numerous fields. For example, in surveillance for detecting people, vehicles or dangerous objects, or in self-driving cars detecting pedestrians, roads, vehicles and other obstacles and signs using the attached cameras in the vehicle.

The object detection process can be typically loosely divided into 4 phases:

- Feature extraction from the image. Using a trained convolutional neural network (CNN), relevant visual features/patterns are extracted from the image and stored in a data structure
- Network predictions. Localization(bounding box) and classification(label) predictions are made from the extracted features. Two-stage object detectors make region proposals from the image first, and then these regions of interest(RoIs) are classified. Single-stage detectors don't use region proposals, instead they make the bounding box and label predictions straight from the extracted features.
- Non-maximum suppression (NMS). Often the object detection model finds multiple

slightly different bounding boxes/RoIs for the same object in the image. NMS is a post-processing method designed to remove these overlapping boxes and leave just one combined bounding box.

- Evaluation metrics. Finally the detection process needs to be evaluated somehow, for this object detection has different popular metrics like mean average precision (mAP) and intersection over union (IoU), which we will discuss further in this chapter. [10]

Object detection is a multi-task problem, since it needs to predict the coordinates of the bounding box and the class of the object at the same time. For this we need to use multi-task learning, which typically means that we have one branch for the coordinates and another branch for the classes at the end of the network. From both of these layers we get parameters to our loss function. When training a model, the aim is to minimize the loss function, because the loss function gives you the difference between the ground truth values of the image and the network's predicted values. [48]

Before neural network based approaches became the norm, there were traditional non-neural methods like HOG [5], SIFT [45] and Viola-Jones [67]. Nowadays, almost every object detection method uses a convolutional neural network, that has been pretrained with some popular dataset, like MS COCO [39] or ImageNet [6], for feature extraction from the input image. One reason to use CNNs is that they provide end-to-end detection with automatic feature extraction, without the need for handcrafted feature selection that needs to be done with the traditional methods [10].

The neural network based object detection approaches can be broadly divided into: two-stage methods that use region proposal and single-stage methods that use regression [50, 10]. The most significant networks that are often used as the base of today's state of the art models will be discussed later in this chapter.

### 2.1.1 Evaluation metrics

There are 2 main performance metrics in object detection: mean Average Precision and Frames Per Second (FPS) [10]. FPS is quite straightforward, it tells you the number of images/frames that the network can detect in a second, so it measures the speed of the network, which is a really important metric when discussing real-time systems. FPS is usually measured over a set of images, by dividing the number of images with the total time that went into processing them. Of course FPS is also highly dependent on hardware architecture and software implementation, so that should be taken into account when comparing results between different environments.

To measure the accuracy of a network we have the mAP. The Precision-Recall Curve (PR Curve) and IoU terms need to be explained first. In object detection IoU is used

to calculate the relative overlap between the ground truth bounding box and a predicted bounding box, so it is a metric for the localization task [48, 60]. Visualization of IoU can be seen in Figure 2.1

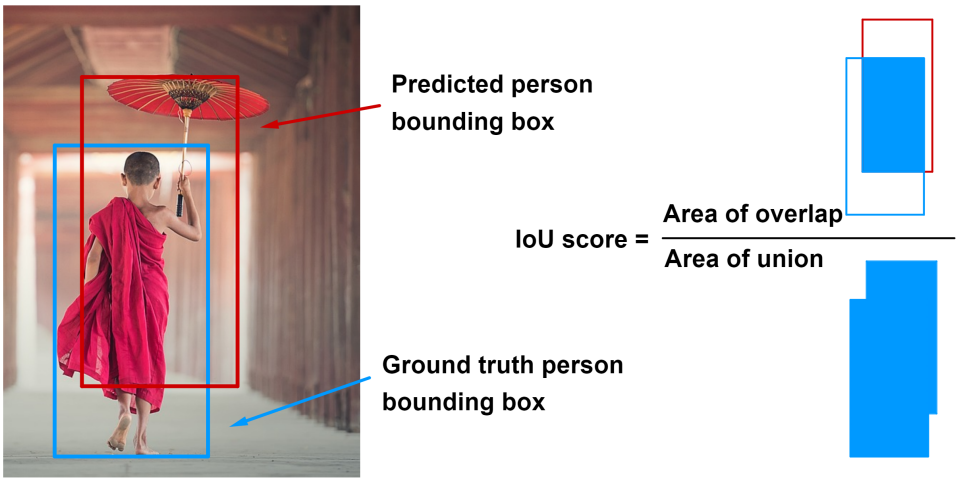


***Figure 2.1.*** *The overlap between the predicted bounding box and ground truth bounding box divided by their combined area is the IoU score. Adapted from [10].*

In Formula (2.1), where $B$ is a bounding box, it can be seen that the maximum value for IoU is 1 and the minimum is 0. IoU value of 1 means that the bounding boxes are identical and 0 means that they do not overlap at all [10]

$$IoU = \frac{B_{\text{ground truth}} \cap B_{\text{prediction}}}{B_{\text{ground truth}} \cup B_{\text{prediction}}} = \frac{\text{Area of intersection}}{\text{Area of union}}. \tag{2.1}$$

A threshold value is used for the value of IoU to determine if a predicted box is a True Positive (TP) or a False Positive (FP). A False Negative (FN) occurs when the network does not detect a ground truth object from the image. When evaluating, the popular MS COCO dataset uses IoU threshold values from 0.50 to 0.95, with 0.05 incrementations and averages out the results [39].

With these detection definitions the Formula (2.2) shows how Precision is calculated and Formula (2.3) shows how Recall is calculated [10]

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{2.2}$$

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{2.3}$$

Precision gives the ratio of true detections to all detections [10]. So if there are a lot of detections by the network where there are no object in the image(FPs), then precision gets closer to zero. In an opposite way Recall gives the ratio of true detections to the total

amount of ground truth objects [10]. So if there are a lot of missed objects by the network, then recall gets closer to zero.

Object detection methods output a confidence score from 0 to 1 for every detection. By changing the confidence threshold we can get different values for Precision and Recall and create a PR Curve. Typically the PR Curve starts from Precision being close to 1 as Recall is close to 0, because the confidence threshold is high and there are no FPs. When the confidence threshold gets smaller, the value of Recall starts rising and Precision starts to come down, as can be seen from Figure 2.2.

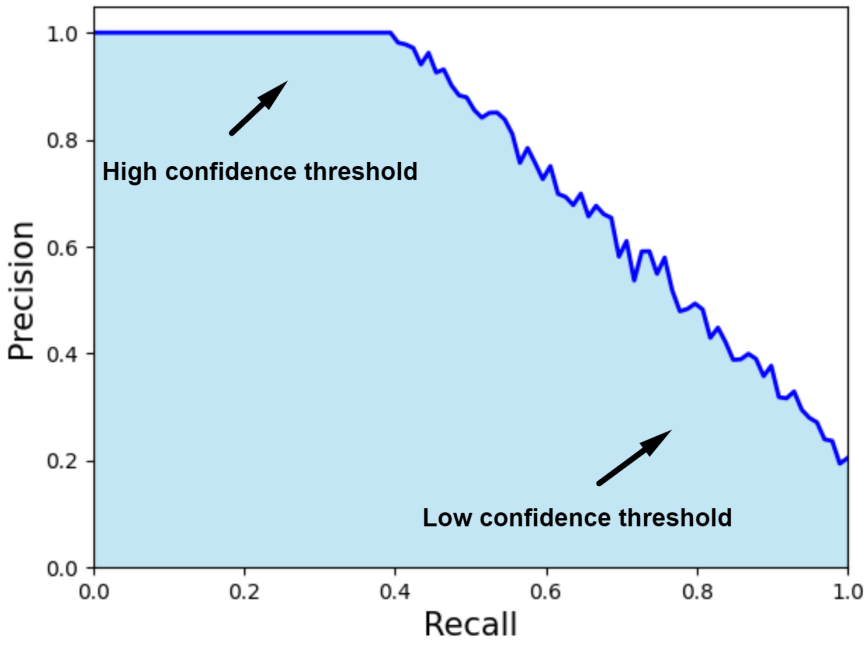


***Figure 2.2.*** *The accuracy of an object detector can be evaluated by using the Precision-Recall curve. Adapted from [10].*

The Average Precision (AP) can now be calculated from the area under the PR Curve. This means that the higher Precision stays as Recall rises the better the AP value of the network is. Now to tie all of this back to mAP, the average AP value over all of the classes combined is the mAP. [10]

### 2.1.2 Non-maximum suppression

Object detection algorithms usually end up outputting multiple slightly different bounding box predictions for the same object in the image, this can be seen in Figure 2.3. For this problem a post-processing technique called Non-maximum suppression is used [10, 48]. NMS is designed to leave only the best bounding box prediction for each object in the image.

Object detectors typically output the class predictions as a percentage confidence value.

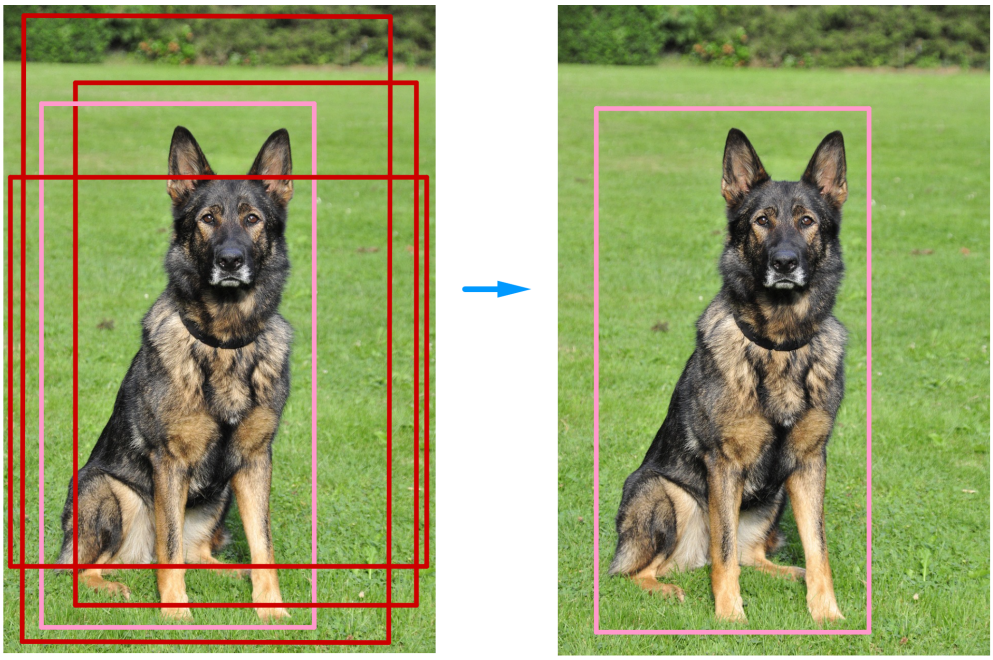


***Figure 2.3.*** *Bounding box predictions before and after applying non-maximum suppression. Adapted from [10].*

For example, if a dataset has three classes: "dog", "cat" and "person", then the output of the classification part of the network could be: [80.0, 15.0, 5.0], which would mean that this bounding box is predicted as a dog at an 80.0% confidence. This is how NMS typically works:

- Remove detections that have a confidence value under a chosen threshold, like 0.4.
- Take the highest confidence bounding box and compare the IoU value of all other same class bounding boxes to that, if the IoU value is above a chosen threshold, like 0.5, then the lower confidence bounding box is overlapping and should be removed from the output list.
- Repeat the previous point with the next highest confidence bounding box until you have gone through all of them. [10, 48]

After running NMS you should only have one bounding box for every object in the image, but you may have to tinker with the confidence and IoU thresholds, depending on the detector's precision.

### 2.1.3 Loss function

When training neural networks, the aim is to minimize the loss function, because it shows the difference between the ground truth values and the predicted values. In object de-

tection, the loss function can differ between network architectures, but generally it always contains localization and classification components. For example, the loss function can often be comprised of 3 parts: Localization loss, Confidence loss and Classification loss [48]. The original YOLO object detection architecture paper follows this style of loss function [58].

The Localization loss is the error between the ground truth bounding box coordinates and the predicted bounding box coordinates. The typical format for horizontal bounding boxes is $(x, y, w, h)$, where $x$ and $y$ are the coordinates for the center point of the bounding box and $w$ and $h$ are the width and height of the box. In Formula (2.4) the first term gets the error between the center coordinates of the bounding boxes [48, 58]. The second term gets the error of the width and height difference of the bounding boxes

$$\begin{aligned} &\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ &+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]. \end{aligned} \tag{2.4}$$

In YOLO the image is divided into a grid, where each small part of the image is called a cell. $S^2$ represents the cells that the image has been divided into and $B$ has all the predicted bounding boxes. The value of $I_{ij}^{obj}$ is 1 when there is an ground truth object in the cell $i$ and the $j$th bounding box has the largest confidence value in the cell $i$ and thus is "responsible" for that prediction. [48, 58]

The Classification loss is simpler, as can be seen from Formula (2.5)

$$\sum_{i=0}^{S^2} I_i^{obj} \sum_{c \in classes} \left( p_i(c) - \hat{p}_i(c) \right)^2, \tag{2.5}$$

where $\hat{p}_i(c)$ is the probability of having class $c$ in cell $i$. The value of $I_i^{obj}$ is always 1 if there is an object in cell $i$, if not, then it is 0. So the more confident the detector is about the classes in the cells, the smaller the loss. [48, 58]

Lastly we have the Confidence (sometimes Objectness) loss, which has 2 terms. The first one checks the confidence loss of bounding boxes that are inside a cell/grid that has a ground truth object inside of them. In Formula (2.6), $\hat{\mathrm{C}}_{\mathrm{i}}$ is the confidence(or objectness) value of bounding box $j$ being in cell $i$. To combat the model making wrong predictions in cells that don't have ground truth object in them, in the second term the value of $I_{ij}^{noobj}$ is 1, when there are no objects in the cell $i$, thus the term increases loss if there are predictions in that cell [48, 58]

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}\left(\mathrm{C_i}-\hat{\mathrm{C}}_\mathrm{i}\right)^2$$
$$+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}\left(\mathrm{C_i}-\hat{\mathrm{C}}_\mathrm{i}\right)^2. \quad (2.6)$$

The sum of all of these losses is the total loss function, which is used to train the detector [58]. These formulas are just one example of a loss function, even the loss functions of the latest YOLO versions differ partially from this original one [57, 3].

### 2.1.4 Two-stage detectors

In object detection Two-stage detectors(sometimes Region proposal-based detectors) have this name, because they divide the object detection pipeline into 2 parts, which are region proposals for localizing objects and then classification, that is based on the proposed regions. Typically two-stage detectors are seen as noticeably slower but possibly more precise compared to single-stage detectors [10]. This means that technologies that need real-time object detection, like self-driving cars, are not generally the place for using two-stage detectors.

The R-CNN family of object detectors are the most well known examples of two-stage detectors and their influence can be seen in the inner workings of state of the art two-stage detectors of today. R-CNN is short for Region-Based Convolutional Neural Network [10].

#### R-CNN

The original paper for R-CNN was released in 2014, by Ross Girshick et al. [15]. At it's proposal date, R-CNN provided state of the art results and pioneered in neural network based methods for object detection.

The region proposal part of the method is done with an algorithm called selective search, which has nothing to do with neural networks and is just a classical image segmentation algorithm [64]. The job of selective search is to find regions from the image that have an object inside them. After extracting these regions of interest, the next step in the R-CNN method is to extract features with a standard pretrained CNN and then classify the objects with an Support Vector Machine(SVM) classifier. The images that are fed to the CNN need to be of a fixed size, so the RoIs are warped to enable this. Along with the object classification, the region proposal coordinates are refined with bounding box regression to get a better representation of the objects with the bounding boxes. [15, 10,

50]

So there are three different modules/classifiers to train: CNN feature extractor, SVM classifier and bounding box regressors. Figure 2.4 shows an overview of the R-CNN architecture.

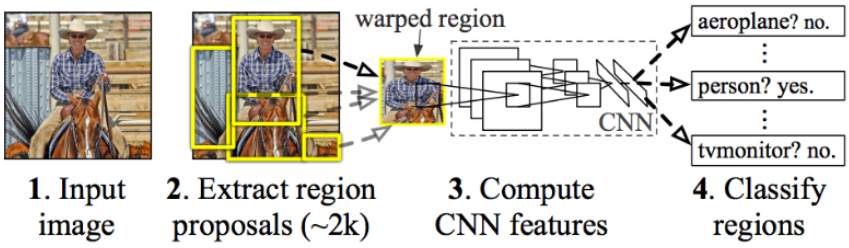


***Figure 2.4.*** *R-CNN architecture. From [15].*

This many modules makes training non-trivial and R-CNN can not be really called an end-to-end method, since it is mainly a combination of existing components that all need to be trained, except for the selective search algorithm [60].

#### Fast R-CNN

In 2015 Girshick improved on his method and called it Fast R-CNN [14]. In Fast R-CNN the order of the first stages is changed so that the feature extraction is done first to the whole image with the CNN, and after that the regions of interest are extracted from the feature maps using a new component called RoI pooling layer [10, 60, 50]. This way the CNN needs to be ran only once and not for every region separately. The SVMs for every class are also replaced by an softmax layer at the end of the network [10, 60].

These changes make Fast R-CNN easier to train since now everything expect selective search happens in the same network. The accuracy and speed are improved over the original R-CNN, but the need for selective search still keeps Fast R-CNN far from real-time object detection [10, 60].

#### Faster R-CNN

The main change in the next iteration of R-CNN is to change the selective search algorithm to a Region Proposal Network (RPN), which means the deep learning network is finally fully end-to-end. This version is called Faster R-CNN, and it was proposed in 2016 by Ren et al. [59].

Like selective search, the RPN takes the last feature map as an input and generates the regions of interest from there. In addition to a fully convolutional layer, the RPN has a binary classifier for determining if the region has an object or not (objectness score),

and a bounding box regressor for pinpointing the region. Outside of the RPN, Faster R-CNN has basically the same architecture as Fast R-CNN. Figure 2.5 visualises the Faster R-CNN detection pipeline. [48]

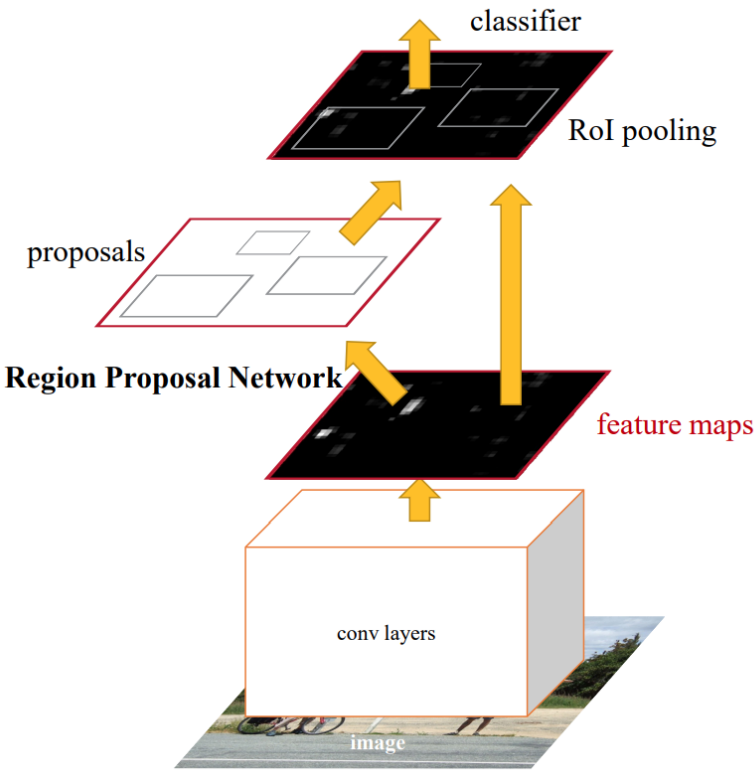


***Figure 2.5.*** *Faster R-CNN: First, feature extraction takes place, followed by extracting region proposals using the RPN, then RoI pooling for each region proposal, and finally, classification and bounding box regression occur at the end. From [59].*

Anchor boxes were also introduced in this paper, which is significant since anchor boxes are used in many modern object detection methods, like SSD and YOLO [50, 41, 58]. RPN divides the image into constant size sections using a sliding window technique [10]. In the center of these windows is the anchor, and from that point RPN generates 9 differently shaped anchor boxes, that have different aspect ratios and scales for every anchor location in the image [10, 59]. The anchor boxes are then used for the objectness scoring and bounding box regression. Faster R-CNN is a huge improvement speed-wise compared to R-CNN and Fast R-CNN. The FPS difference can be as much as 200% compared to the original R-CNN [50, 10, 48].

### 2.1.5 Single-stage detectors

Single-stage detectors have this name, because they mold the localization problem and the classification problem in such a way, that the results can be acquired in a single pass of a network. This simpler structure makes for faster inference times, but may lead to

slight drops in precision. [10, 50]

Single-stage detectors, SSD and YOLO, were proposed a little after the R-CNN family of detectors, and quickly became popular, thanks to their big FPS improvement and similar precision results. These faster detection times make real-time object detection with acceptable precision possible.

### SSD

The Single Shot Detector(SSD) was proposed in 2016, by Wei Liu et al. It achieved comparable precision results and outperformed Faster R-CNN in terms of FPS [41, 10].

The first part in the SSD network architecture is a pretrained base network for extracting the feature maps from the image. In the second part there are 6 convolutional layers that decrease in size. The idea is that the largest convolutional layers have the smallest anchor boxes in relation to the image, so they are able to detect the smallest objects in the image. On the other hand the smallest convolutional layers have the largest anchor boxes, so large objects are also detected. This way detections of objects of all sizes are acquired. All of these detections are sent to the finals phase: NMS, which tries to leave only one bounding box detection for every object. [10]

Unlike YOLO or Faster R-CNN, SSD does not predict the objectness value for a given bounding box, instead it predicts the class probability straight away [50]. This means that there is always a pre-determined amount of detections that reach the NMS stage, which filters out the redundant detections.

### YOLO

YOLO, short for You Only Look Once, is a family of object detectors, like R-CNN. The original YOLO paper was published by Redmon et al. in 2016 [58]. It is a competitor for the SSD, since they both are single-stage detectors and much faster than Faster R-CNN.

YOLO treats object detection as a single regression problem, which means it is able to predict the bounding boxes and class probabilities of an image in a single pass through the neural network. Figure 2.6 visualises the detection pipeline. In YOLO, the input image is split into a grid of cells $S$x$S$. Each cell predicts a defined number of bounding boxes and for every prediction, YOLO carries the bounding box coordinates and the objectness/confidence score. The class probability predictions are calculated for each cell. Finally, again in the end NMS gets rid off the worse predictions. [58]

The first three versions of the YOLO family are considered the originals, since they were all published by Redmon et al. [58, 56, 57]. The first YOLO version makes a notable amount of localization errors and has a low recall value, when compared to region

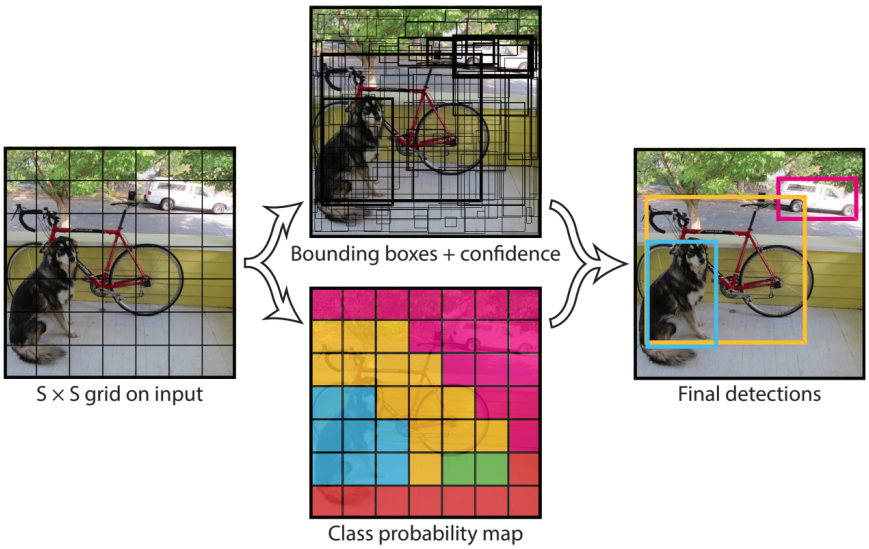


***Figure 2.6.*** *YOLO detection pipeline. From [58].*

proposal-based methods like Fast R-CNN [58].

YOLOv2(2016) tries to improve on those two aspects in multiple ways, one of which is to use the anchor boxes, which are introduced in Faster R-CNN. The anchor boxes are chosen by doing k-means clustering to the dataset [56]. This leads to over 10 times more bounding box detections and helps with localization and recall [56].

For YOLOv3(2018), one of the bigger changes is taking the anchor-boxes further by making object detections in three different scales, and at every scale, a cell has three different anchors [57]. So in total a cell has 9 anchors to predict with in one pass of the network. This multi-scale detection is again for detecting smaller and larger objects, like in SSD.

All the first three YOLO methods use an open-source neural network framework developed in C by Redmon, which is called Darknet [58]. YOLOv2 uses a 19 layer CNN backbone for the base feature extracting called Darknet-19 and YOLOv3 uses a much deeper Darknet-53, which uses 53 layers and significantly increases the capacity of the model, but is slightly slower [56, 57].

Many new YOLO versions have been published over the years. YOLOv4(2020) still uses the Darknet framework and introduces a multitude of data augmentation methods and training strategies [3]. A company called Ultralytics has released YOLOv5(2020) and YOLOv8(2023), which are built on PyTorch framework instead of Darknet. Ultralytics provides a versatile and easy to use open-source repository for using those methods [27]. YOLOX(2021) is the first anchor-free version of YOLO after the first one, which might be the future for YOLO methods, as the currently latest version YOLOv8 also does not use anchor boxes [13, 28].

## 2.2 Oriented object detection

The localization part of object detection generally uses horizontally aligned bounding boxes(HBBs) to represent the location of the object in the image. Common representation for these HBBs is ($x$, $y$, $w$, $h$), where $x$ and $y$ are coordinates for the center point of the bounding box and $w$ and $h$ represent the width and height of the bounding box. Using horizontal boxes can become problematic, when the borders of the bounding box do not tightly fit the orientation of the object, leading to a situation, where a significant part of the area inside the bounding box does not represent the object. To solve this problem, most oriented object detection methods add a parameter for the bounding box representation: ($x$, $y$, $w$, $h$, $\theta$), where $\theta$ denotes the angle that the bounding box deviates from the axis. These can also be called oriented bounding box detection methods. Figure 2.7 demonstrates how oriented bounding boxes(OBBs) can be a better representation for oriented objects in the defined bird's eye view case.

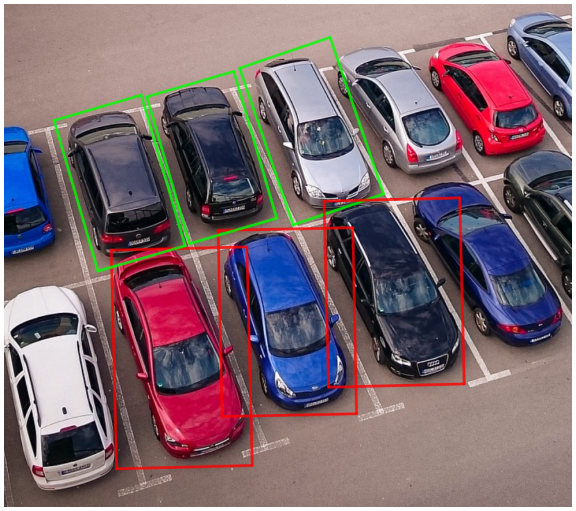

***Figure 2.7.*** *Oriented bounding box marked with green. Horizontal bounding boxes marked with red.*

### 2.2.1 Applications

The two most researched applications for oriented object detection are aerial image and natural-scene text detection. In aerial imagery, this approach is crucial for identifying objects like ships and vehicles, which can be oriented at different angles due to the perspective of the image. Similarly, in natural-scene text detection, accurately recognizing and understanding text that appears at various orientations is important for effective optical character recognition(OCR) and text extraction.

Aerial images or in other words: Remote sensing images(RSIs) offer a different challenge compared to natural-scene object detection. This is largely the consequence of three factors, that are also demonstrated in Figure 2.8:

- Arbitrarily oriented objects: Ships, planes and other objects can be in various orientations. Detectors need to be able to localize the objects without overlapping them.
- Huge aspect ratio and scale differences: Objects like bridges and football fields are significantly larger than smaller objects like cars and trucks, which means that the detector needs to detect different size objects and a huge scale of aspect ratios.
- Dense objects. Objects like vehicles and boats are often very densely packed in RSIs. They can also be just a few pixels in size. Distinguishing these objects from each other can often be difficult for detectors.

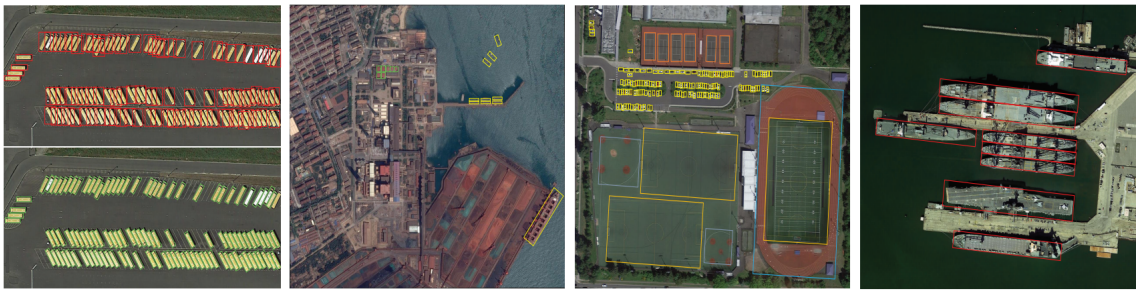


***Figure 2.8.*** *RSI examples from the HRSC2016 and DOTA datasets. (a) The arbitarily oriented objects can be better represented with OBBs (bottom) compared to the HBBs (top). (b) RSIs can have a complex background, so objects are difficult to detect (c) The scale of different objects can vary significantly. (d) Many objects, like ships, can have big aspect ratios. From [22].*

RSI-based object detection is the most researched application in the domain of oriented object detection in recent years, partly due to it's many potential use cases, like urban management [61], precision agriculture [55, 62] and emergency rescue [51]. Research has also seen a rise since the Large-scale Dataset for Object Detection in Aerial Images(DOTA) was introduced in 2017 [72]. It had much more images, classes and instances compared to others similar datasets. Futhermore DOTA has more recently had some updates, and it now has 4 times the amount of aerial images compared to the original release (11268 images) [8]. For these reasons DOTA is generally the main benchmark in all aerial image object detection papers.

Understanding text from natural scenes is usually divided into two parts in computer vision: text detection and text recognition. The detection part is about localizing the texts from the image, so it is the more relevant part for this thesis. Natural-scene text detection is another, nowadays less researched, application for oriented object detection. Like RSIs, natural scene images can have big or small texts with big differences in aspect ratios and arbitrary orientations, which is why HBB-based methods are not always good enough for detection, and the use of OBBs or quadrilaterals is a more accurate way of representing the texts in these images [47, 35, 36, 42, 92]. Figure 2.9 shows an example

from the R$^2$CNN paper, which is a scene text detection method, of the inadequacy of using HBBs for localization in natural-scene text detection.

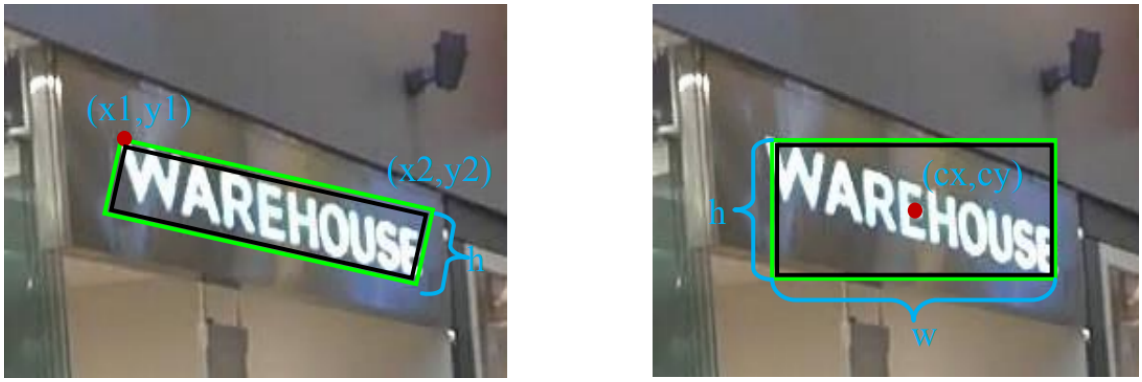


***Figure 2.9.*** *Oriented bounding box on the left. Horizontal bounding box on the right. From [26].*

Compared to traditional optical character recognition methods, the lighting conditions, fonts and perspectives in natural-scene text detection are generally uncontrollable, making it a difficult detection problem [35, 26]. Detecting the rotation of a text in an image creates the possibility to rotate the text-OBB so that the letters are upright, which can make recognizing the letters easier. Applications for natural-scene text detection include: License plate recognition [19], road sign recognition [16], photo translation [1]. The most popular datasets for natural-scene text detection are the ICDAR [29] and COCO-text [66] datasets.

### 2.2.2 Key attributes

Oriented object detection methods can be divided into four different categories based on their most defining features:

- **Anchor-based methods** use the aforementioned anchor-boxes to localize the regions of interest from the image. In the oriented object detection case, either normal HBBs are used for the anchors as references and OBBs are generated from those usually with bounding box regression [26, 83, 81], or a lot more different anchor-boxes are designed and used to cover all the possible object orientations that are potentially present in the image [47, 89].

  The first solution has many drawbacks like the problem of possible huge differences between the horizontal anchor box and the oriented ground truth box that is the regression target, which leads to decreased robustness [4]. The second solution tries to add oriented anchor-boxes, which leads to increased computation times when proposing regions, slowing the whole detection process down.

  Despite the drawbacks, most oriented object detectors currently still are anchor-

based methods [4, 22, 17].

- **Anchor-free methods** have gained popularity in object detection research in recent years, because they aim to get rid of the drawbacks and restrictions that the anchor-boxes bring. These anchor-free object detection methods are mostly keypoint-based [30, 91, 9, 63, 86], which means they aim to detect keypoints like corners, extreme points or the center of an object and then regress the bounding boxes and labels from them. For example the CornerNet (2018), detects the top-left corner and the bottom-right corner [30].

  While some of these anchor-free methods have been adapted to work with oriented bounding boxes [71, 73, 31], the keypoint-based methods are often based on single-stage frameworks and currently their performance is relatively limited [87].

- **Angle-based methods** use the common 5 parameter ($x$, $y$, $w$, $h$, $\theta$) representation for the oriented bounding boxes. Most state-of-the-art oriented object detectors are angle-based and thus use this representation [7, 18, 26, 74, 83, 17, 47, 81, 77, 89], but the $\theta$ angle parameter introduces an issue called: boundary discontinuity problem, which has to do with the periodicity of angles and negatively affects the localization accuracy of the detector [79, 84, 77].

- **Angle-free methods** generally use the 8 parameter ($x1$, $y1$, $x2$, $y2$, $x3$, $y3$, $x4$, $y4$) representation for the OBBs, so it has $x$ and $y$ coordinates for every corner of the bounding box. Current angle-free detectors perform quadrilateral regression directly, to find the bounding box [90, 78]. This is a less complex way compared to the angle-based representation, and the boundary discontinuity problem is not a problem anymore, since there is no parameter for the angle.

  Unfortunately the detection performance of angle-free methods is currently inferior compared to angle- and anchor-based methods [90], but a lot of research is currently being done on angle-free methods [76, 53, 90, 22, 31, 88].

### 2.2.3 Boundary discontinuity problem

Angle-based oriented object detection methods introduce a boundary discontinuity problem [79, 84, 77]. This problem occurs as we try to regress the angle value of the fifth parameter and it is caused by the periodicity of angles. A slightly simplified example: The apparent angle difference of an OBB that is tilted 5° and another that is tilted 175° is just 10°. So they look similar, but the integer value that we have to regress has a difference of 170. This means that the discontinuity of values at the 180°-0° boundary will affect negatively to the training, because those OBBs that have a rotation degree value around the boundary, will either have a really large or really small ground truth value, which will meddle with the training process and limit performance.

To give a more practical and realistic idea of the boundary discontinuity problem, we can take for an example the situation in Figure 2.10, where the ($x$, $y$, $w$, $h$, $\theta$) representation is in use, and where $w$ and $h$ represent the shorter and the longer edge of the bounding box respectively. In this case, the angle of the OBB is calculated between the horizontal x-axis and the shorter edge. If there is a slight change in the edge length of an almost square bounding box, the longer and shorter edge can swap places, thus moving the angle $\theta$ value by 90°. Because the aim is to regress the angle value to an integer value, this $\theta$ value change of 90 degrees between similar bounding boxes will again confuse the training process.[90]

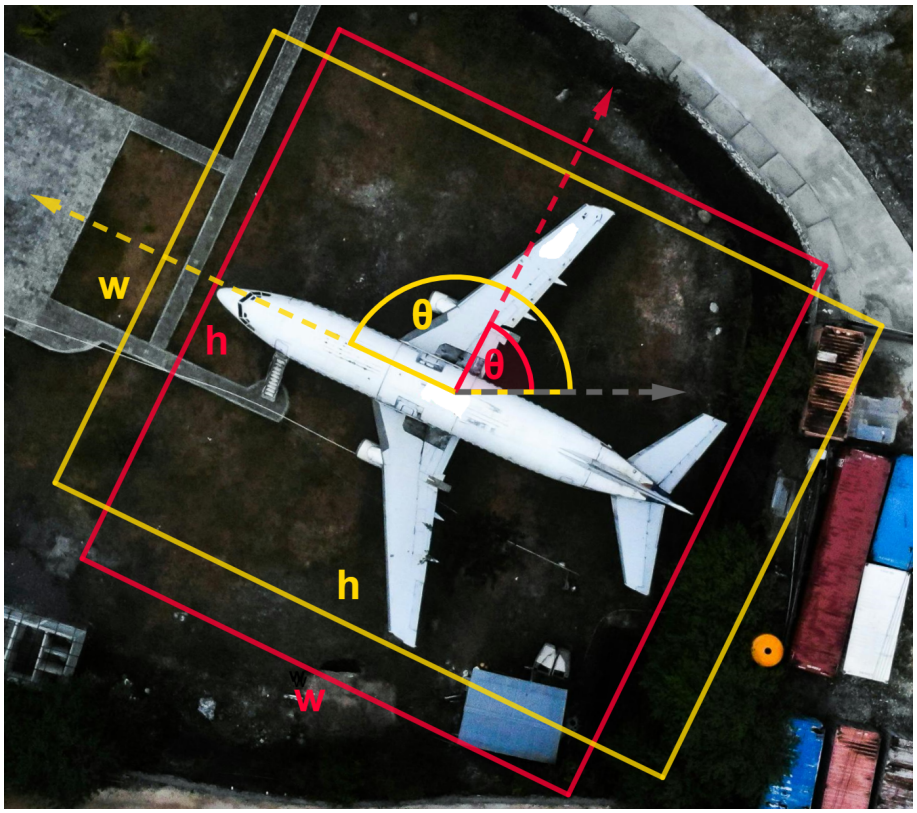


***Figure 2.10.*** *Boundary discontinuity problem of angle prediction. Red and yellow bounding boxes are similar, but they have very different angles, due to the fact that their longer and shorter edges are not on the same side. Adapted from [90].*

A small error in angle prediction can cause the IoU between the ground truth and the predicted OBB to be significantly smaller, which is why many different approached have been developed to handle this problem. SCRDet tries to alleviate the problem by applying a constant IoU factor to the smooth L1 loss function that is defined in Fast R-CNN, which negates the sudden rise in loss in the boundary situation [83]. Yang et al. created circular smooth labels(CSL) to transform the regression task of the angle into a classification task, which then consequently removes the whole problem that comes with regressing the angle [77, 78]. The CSL approach has seen use in other papers [54, 69, 79]. Angle-free methods like Gliding Vertex and RSDet use the eight-parameter quadrilateral regression

to get rid of angular periodicity [76, 53]. There are also some other creative ways to get rid of the angles, and they are gathering more attention from researchers recently [90, 22, 31, 88].

### 2.2.4 Oriented bounding box definitions

Angle-based oriented object detection requires OBBs for training the network. In order for the network to learn the angle parameter in training, and to push out the correct angle while detecting, an OBB definition must be agreed upon. There are three popular OBB definitons: The OpenCV definition(oc), the 90° Long Edge definition(le90) and the 135° Long Edge definition(le135) [93, 78].

The OpenCV definition comes from OpenCV's minAreaRect function, which finds the minimum area rotated rectangle for a 2D point set and returns the ($x$, $y$, $w$, $h$, $\theta$) parameters [52]. OpenCV changed the definition in OpenCV version 4.5.1, so now there exists an old oc and a new oc [93]. Both oc's $\theta$ parameter is the angle between the horizontal x-axis and the width side of the rectangle. The old oc gives $\theta$ values between -90° and 0°, This angle range of a quarter of a circle is made possible, by interchanging the naming of the width and height sides of the rectangle with each other, when going over the -90° limit. This allows the angle parameter to return to again start from 0°, because the angle between the horizontal x-axis and the width side of the rectangle is now different. In a possibly more intuitive way, while the horizontal x-axis is to the right and the vertical y-axis is down, the width side, from where the angle is calculated, is the side between the highest point of the rectangle and the point to the right from it. This can be seen from Figure 2.11 a. In the case where there are two highest points, then the rightmost of them is chosen and the width side is the rightmost vertical side, resulting in a -90° angle. Now the new oc switches the range to go from 0° to the positive 90°, which just means that, as the y-axis is down, now the width side is between the lowest point and the point to the right from it. And again if there are two lowest points, then the rightmost of them is chosen, and thus the rightmost vertical side is the width. The difference on how to acquire similarly rotated bounding box between the old an the new OpenCV definition can be seen from Figure 2.11.

The Long Edge definitions, as the name suggests, calculate the angle between the longest side of the bounding box and the horizontal x-axis. The longest side is also assigned as the width of the OBB similarly to the OpenCV definition. Both Long Edge Definitions use a 180° range for the angle to present all the possible orientations of the bounding box. The le90 definition uses a range from -90° to 90° and the le135 definition uses a range from -45° to 135° [93]. The difference between the definitions ca be seen in Figure 2.12. In Figure 2.12 b. the 120° angle is used to demonstrate, that the le90 definition can not use that, but a -60° angle instead. The -30° representation on the contrary is

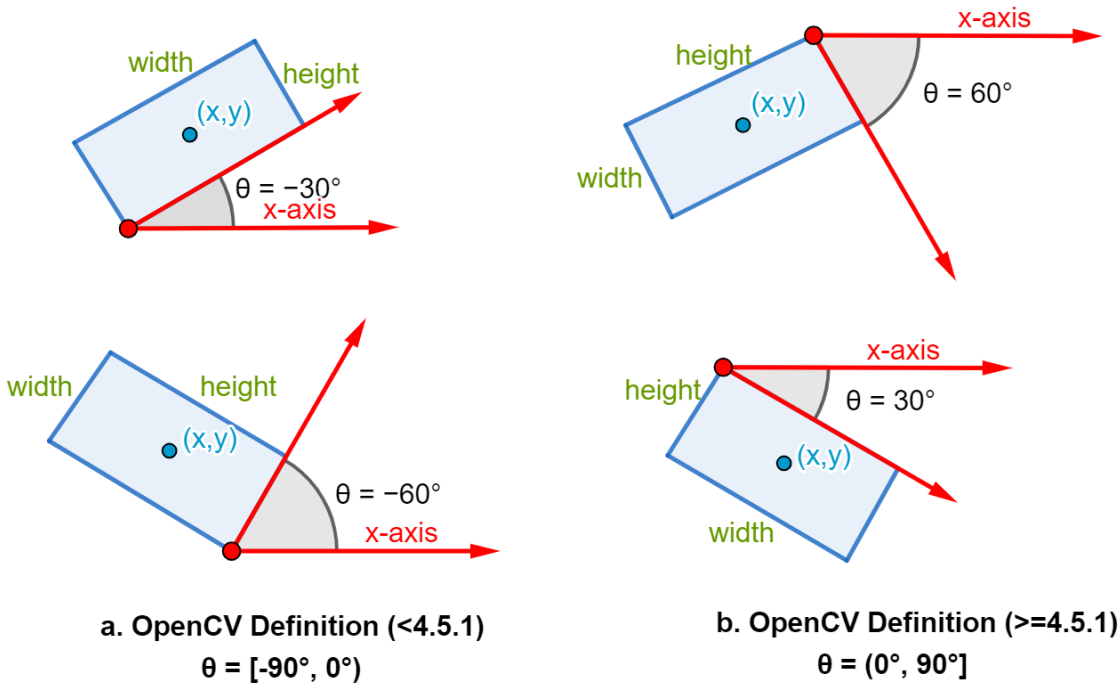


***Figure 2.11.*** *Two examples of the old and new OpenCV OBB definitions. Adapted from [93]*

the same for both definitions.

As to what is the meaningful difference between these two definitions performance-wise could not be found in the literature, but one could assume that is has to do with the fact that the le90 angle-boundary is at the -90° position, which is the horizontal position for the bounding box, which is a very popular orientation for an object in an image. And for this reason one could argue that the le135 angle-boundaries are better, because not as many objects will affect or get effected by the boundary discontinuity problem.

### 2.2.5 Loss function

Training angle-based detectors often involves specialized loss functions that penalize errors in angle prediction or bounding box orientation. Skew IoU appears to be one of the first loss functions to take the orientation of the bounding boxes into account [47, 83, 81]. The intersection area is calculated with the use of triangulation, which means that the intersection area is divided into triangular areas by using the overlapping vertices, and then the areas are summed together [47].

In addition to the ability to take orientation into account, the advantage of Skew IoU compared to a general loss function is that as the aspect ratio rises, the change in angle has a larger effect on the loss value. The problem with Skew IoU computation is that it is not

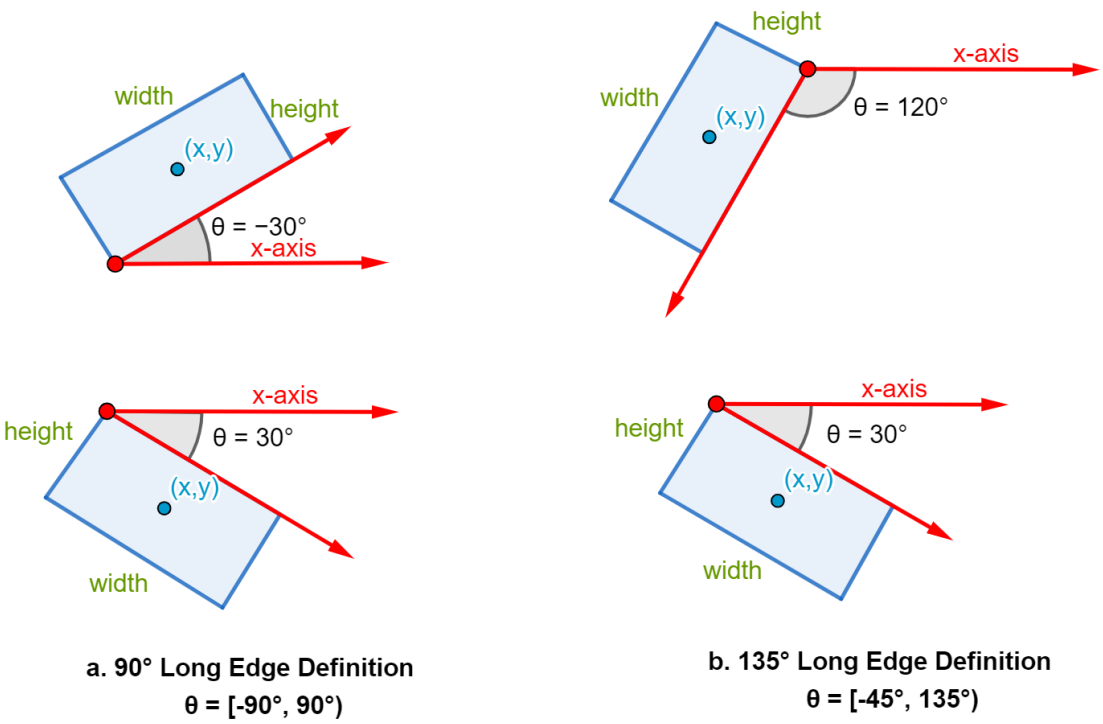


***Figure 2.12.*** *Two exmples of the 90° and 135° Long Edge OBB definitions. Adapted from [93]*

differentiable, which would be needed to perform gradient descent. An approximate Skew IoU loss can still be used to gain some of the benefits. [81]

Four newer losses appear repeatedly in the recent literature:

- Gaussian Wasserstein distance(GWD) loss uses the Wasserstein metric to calculate the distance and takes inspiration from SkewIoU [82, 70].
- Kullback-Leibler Divergence(KLD) loss uses the KLD as the distance [80].
- Kalman Filter IoU(KFIoU) loss uses Kalman Filtering and mimics SkewIoU [85].
- Probabilistic IoU(ProbIoU) loss uses the Bhattacharyya distance [43].

ProbIoU, GWD, KLD and KFIoU are all similarity measures based on Gaussian bounding boxes and they offer a more efficient loss function for oriented bounding boxes. They convert the OBBs to a 2D Gaussian distribution and measure the loss as the distance metric between the 2D Gaussian distributions. [70, 34]

The combination of ProbIoU and Distribution Focal Loss(DFL) for the angle prediction seems to be slightly popular, as it is in use in two newer OBB detection methods [70, 28]. DFL can create a distribution of angles or bounding box offsets [32]. It is a solution to a supposed difficulty in determining the orientation of an OBB, while converting to a Gaussian bounding box, when the OBB is roughly square [70].

# 3. METHODS

In this chapter all the used methods are introduced. The oriented object detection methods were chosen from freely available online repositories. This way, many different methods could be tested, without doing the whole implementation of each research paper from start to finish. Four of the methods used are from the MMRotate [93] open-source toolbox for oriented object detection, one is from PaddleDetection [2] object detection toolkit and one is from a single independent GitHub user [24].

MMRotate is part of the open-source computer vision project called OpenMMLab. MMRotate features a large number of supported methods for oriented object detection, but to be able to compare methods between platforms and evaluate using the same code, the detectors were converted to TensorRT format using MMDeploy, which is a model deployment toolset of OpenMMLab. MMDeploy only supports converting six MMRotate models, of which four are featured in this thesis and the remaining two: Oriented R-CNN and Rotated Faster R-CNN gave out errors when trying to convert them to TensorRT format using MMDeploy, so they are left out of this thesis. [93]

The PaddleDetection object detection toolkit is based on the PaddlePaddle open-source deep learning platform, which is developed by Baidu. Additionally to basic object detection, PaddleDetection offers methods for instance segmentation, multi-object tracking and keypoint detection. It has over 30 object detection methods, but only three of them are designed for oriented object detection, and the one that has achieved the highest mAP is featured in this thesis. This detector model was also converted firstly to ONNX format and then to TensorRT. [2]

The GitHub project called yolov5_obb by Kaixuan Hu is based on the YOLOv5 code by the Ultralytics team [24]. Ultralytics is responsible for releasing v5 and v8 YOLO versions and their open-source implementations and pretrained models on GitHub [27]. The yolov5_obb project also combines code from the CSL paper implementation and the Oriented R-CNN paper implementation, which is modified from MMDetection [78, 74]. Model conversion to TensorRT is supported.

## 3.1 Rotated RetinaNet

In 2017, the Facebook AI Research team, featuring Ross Girshick, known for the development of the popular R-CNN method, published a paper proposing a method called RetinaNet [38]. RetinaNet is composed of three main parts: A ResNet-101 or ResNet-50 backbone neural network, a Feature Pyramid Network(FPN) as the neck of the method and lastly a loss function, called Focal Loss, which is the new novel thing proposed in the paper [37, 20, 38]. RetinaNet is also a single-stage detector, making it the first single-stage detector to beat out all the then-current two-stage detectors on COCO test-dev dataset, while still maintaining better inference times [38]. Rotated RetinaNet is MMRotate's implementation of RetinaNet that is modified to be able to use oriented bounding boxes [93].

The Focal Loss function tries to address the class imbalance problem, which is caused by a large number of easy negatives compared to the amount of hard positives, when training a detector. Easy negatives are RoIs, which the detector easily predicts to not have any objects inside with a high probability, so the confidence score would be something like 0.9. In training, the cross-entropy loss value for confidence of 0.9 is around 0.1, which is close to zero, but it still increases the loss by a tiny bit. The problem becomes apparent when there are so many easy negatives, that the loss is largely affected by the easy negatives, which don't contribute useful learning for the network and can lead to degenerate models [38]. This problem is of course more apparent in single-stage detectors, because they can use thousands of RoIs, because for every grid there is a set of anchor boxes. two-stage detectors propose less regions, and they have a higher chance of having an object inside of them.

The Focal Loss adds a factor to the Cross Entropy loss, which greatly lowers the loss value for predictions that have a high confidence, like easy negatives and positives. This way hard and misclassified examples get more attention [38]. The added factor is $(1 - p_t)^\gamma$, where $p_t$ is prediction confidence and $\gamma$ is the focusing parameter. Having $\gamma \geq 1$ lowers the loss value on predictions that have a high confidence. When $\gamma = 0$, the Focal Loss is the same as Cross Entropy loss. The original paper found $\gamma = 2$ to be the optimal value for the focusing parameter [38]. Figure 3.1 shows the effect on loss with different values of $\gamma$.

The MMRotate Rotated RetinaNet implementation has the most common combination of the key attributes for a oriented object detector, as it is an anchor-based and an angle-based detector. It uses the less deep 50 layer version of ResNet, which should be a good choice since the TPL dataset is very small. The backbone is pretrained with ImageNet-1K dataset [6]. The Feature Pyramid Network is used as the neck.

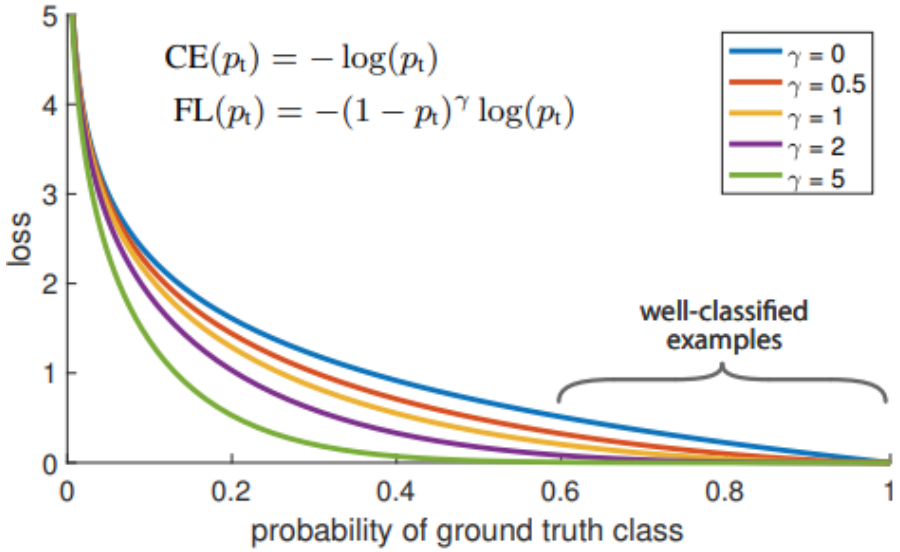


***Figure 3.1.*** *The formulas for Cross Entropy (CE) and Focal Loss (FL), and a plot illustrating the decrease in Focal Loss, when using larger values for the focusing parameter $\gamma$. Well-classified examples are the easy positives/negatives which have high prediction confidence and don't contribute much to learning. From [38].*

## 3.2 RoI Transformer

Wuhan University's CAPTAIN team has made many contributions to oriented object detection, methods like ReDet [18] and $S^2$A-Net [17], and also the popular DOTA dataset [72, 8]. In 2018 they published the RoI Transformer, which has two main parts: A Rotated RoI learner that is able to transform the horizontal RoIs into rotated RoIs (RRoI), and a Rotated Position Sensitive RoI Align (RPS-RoI-Align) module for extracting the features, which are rotation-invariant, from the RoIs [7]. The RoI Transformer aims to fix the problems that usually appear when using oriented anchor boxes, which happen either when too many different scale, angle and aspect ratio anchors are used, leading to increased computational complexity, or when too few anchors are used, leading to low recall values on images having complicated rotated objects [7].

Figure 3.2 shows the RoI Transformer module in the object detection pipeline, other than that the rest of the implementation in the RoI Transformer paper is mostly based on Faster R-CNN using a Region Proposal Network, and thus is considered both anchor-based and a two-stage detector. More specifically the research paper's implementation also takes inspiration and uses the head from Light-Head R-CNN [33] and the neck from FPN [37].

The RRoI Learner takes the predicted horizontal RoI and, using regression, predicts the offset for the RRoI [7]. After the RRoI Learner, RRoI warping takes the feature maps and outputs geometrically robust features, which basically means that all the objects are in the same orientation, which allows for rotation invariant features to be extracted next [7]. The angle-based RoI Transformer module can be added to other existing methods by adding

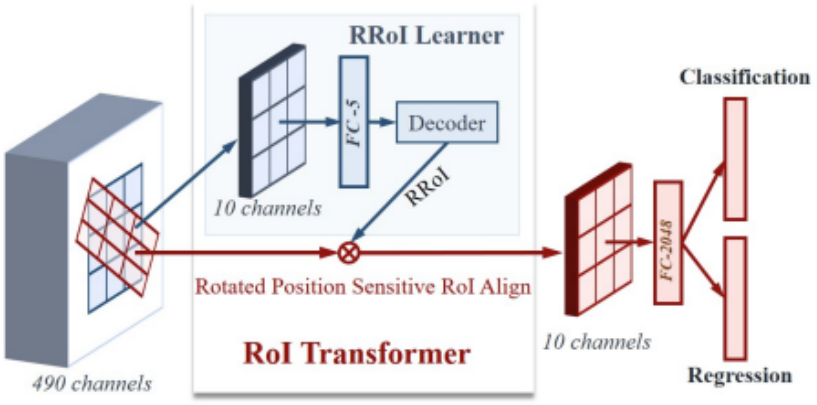


***Figure 3.2.*** *The RoI Transformer module recieves the horizontal RoI, from which the RRoI Learner regresses the offsets by using a fully connected layer and outputs the RRoI. From [7].*

the RoI Learner and replacing the usual horizontal RoI warping part of the pipeline with RRoI warping to provide the Rotated RoIs and achieve better oriented object detection performance.

The MMRotate implementation for RoI Transformer again uses ResNet-50 as a backbone network instead of ResNet-101, which was used in the original paper. The FPN is configured in MMRotate, but there is no mention of Light-Head R-CNN. That may just be a naming convention difference since, there is a head called RoITransRoIHead, which is only used by RoI Transform and ReDet [18] configurations in MMRotate. RPN head is also in use.

## 3.3 Gliding Vertex

Gliding Vertex is another object detection method by the CAPTAIN team [76]. Released in 2019, the Gliding Vertex does not use angle regression, making it an angle-free method solving the boundary discontinuity problem, instead it regresses a quadrilateral by gliding the vertices of each side of the predicted horizontal RoI [76]. So the offset between the corners of the horizontal RoI and the corners of the wanted quadrilateral is the target of the regression. The horizontal RoIs are generated by the Faster R-CNN based RPN, making Gliding Vertex a two-stage and an anchor-based detector.

Oriented object detection methods using OBBs or quadrilateral bounding boxes (QBB) can suffer from the bounding box ambiguity problem [49]. This happens when a oriented object can be represented using multiple bounding boxes that look the same, but have different values for different parameters. Especially the QBBs can have this problem, if every corner point is regressed, then the unordered points have $P_4^4 = 24$ permutations to represent the same bounding box [49]. The representation ambiguity problem leads

to worse network training, since the ground truth bounding box is always the only representation that the prediction is compared to, making the loss value high, if the predicted representation is not the ground truth one.

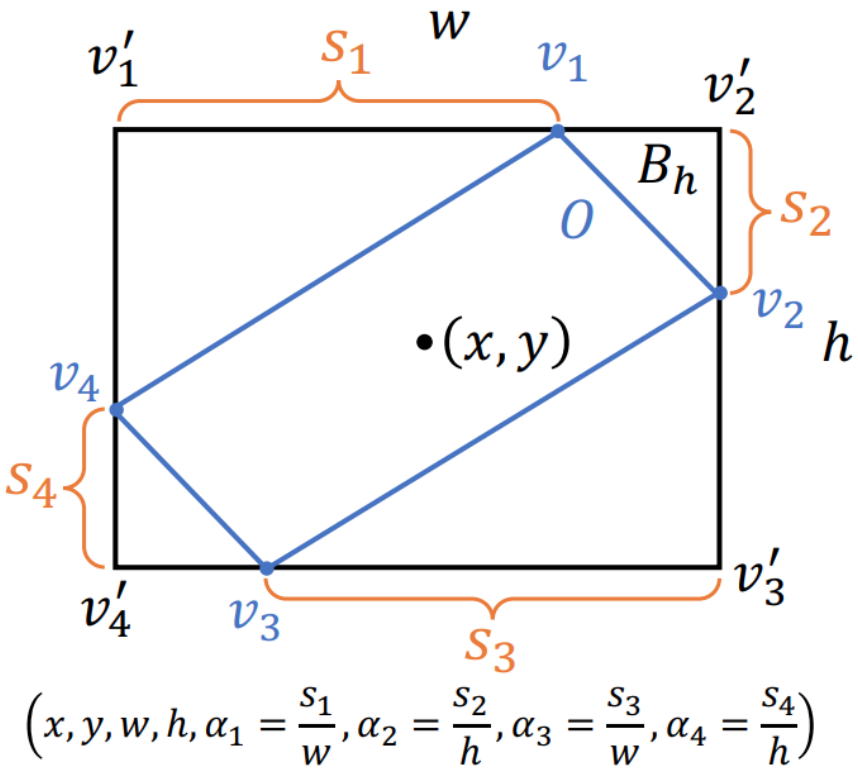


***Figure 3.3.*** *Gliding Vertex bounding box representation illustrated. The black HBB is the initial RoI, from where the $s_i$ values are regressed to rotate the bounding box and end up with the blue OBB. From [76].*

Gliding Vertex addresses the ambiguity problem by the vertex offset representation and thus limiting the values the offset values can have. The proposed representation is shown in Figure 3.3. The bounding box representation is $(x, y, w, h, \alpha_1, \alpha_2, \alpha_3, \alpha_4)$, where each $\alpha$ has it's own corner and the corresponding side, on which they set the position of the final bounding box corner. The $\alpha_i$ values are normalized to [0,1]. Another advantage of this representation compared to the angled one is case of the long and slender object, where a slight error on the angle value plummets the IoU score. But slight changes on the Gliding Vertex representation won't have that effect.

Additionally Gliding Vertex introduces an obliquity factor to handle ambiguity with almost horizontal object orientations. The obliquity factor checks the area ratio between the predicted OBB and the HBB, if the ratio is close enough to one, then the HBB is chosen as the predicted area. The obliquity factor $r$ is the fifth extra variable in the methods output.

The MMRotate implementation uses the ResNet-50 backbone, FPN neck and RPN and Gliding Vertex heads.

## 3.4 RTMDet-R

RTMDet object detector was proposed by Lyu et al. in 2022, with the purpose of competing with YOLO models in object detection, also with oriented object detection in mind [46]. The two main improvements presented in the paper are achieved by optimizing the architecture, and the training strategies of the method.

The architechture of RTMDet takes inspiration from typical single-stage detectors. It uses CSP-blocks as the backbone of the network, which are also used by newer YOLO methods, like YOLOX and YOLOv4 [13, 3].

The CSP-blocks are from the Cross Stage Partial Network (CSPNet), which was introduced in 2019 by Wang et al. CSP divides the base layer feature map into two parts, of which the other part goes through the dense layers and then the two parts are merged together. The aim is help the information flow in the network and to reuse features that are extracted in earlier stages, which removes redundant computations. This makes the network more lightweight, which increases inference speed, while maintaining or improving accuracy. CSP can be applied to all kinds of backbone networks to gain these benefits. [68]

The neck has a PAFPN-block, which is a combination of a FPN and a Path Aggregation Network(PAN) [46]. The CSP-block is divided into three scales, which the PAFPN takes and uses both top-down and bottom-up feature propagation to improve feature mapping, enhancing performance. PAN was released in 2018 by Liu et al. for image segmentation [40]. PAN uses the multi-scale output of FPN to do bottom-up path augmentation in a pyramid-like way, to shorten the information path between the lower and higher layers using skip connections. Additionally PAN uses Adaptive feature pooling, which takes useful features from all of the PAN layers to make detection proposals. PAN makes detection more accurate, while maintaining inference speeds [40].

The RTMDet paper does not describe how its anchor-freeness is achieved, but considering how YOLOX clearly seems to be the main inspiration, it is safe to assume that a similar technique is used. YOLOX divides the image into a grid, and uses the top left corner of every grid-cell as an anchor-point [13]. The aim is to find the anchor point that is closest to the center of the object and then regress the four bounding box parameters $(x, y, w, h)$ from that anchor point. The image is also divided into three different scaled grids. In training, in addition to the object center point, a 3x3 area of points are assigned as positives, which should balance the amount of positive and negative samples.

RTMDet aims to optimize the parameter-accuracy trade-off by balancing the resolution, depth and width of the backbone and neck. Figure 3.4 shows RTMDet beating the other YOLO-based detectors in this aspect. The less parameters a method has usually equates to faster inference times. Modifying the basic CSPDarkNet backbone, RTMDet proposes

using a large-kernel 5x5 depth-wise convolution layers, which help the method to get a better global context from the image, and thus increases accuracy [46]. The backbone is also widened and the number of blocks is reduced to improve inference speed while maintaining accuracy. The backbone and neck are balanced, by moving computations from the backbone to the neck, by adding depth to the basic blocks of the neck, thus making the backbone and neck more similar and increasing the capacity(ability to learn complex patterns) of the Feature Pyramid [46].

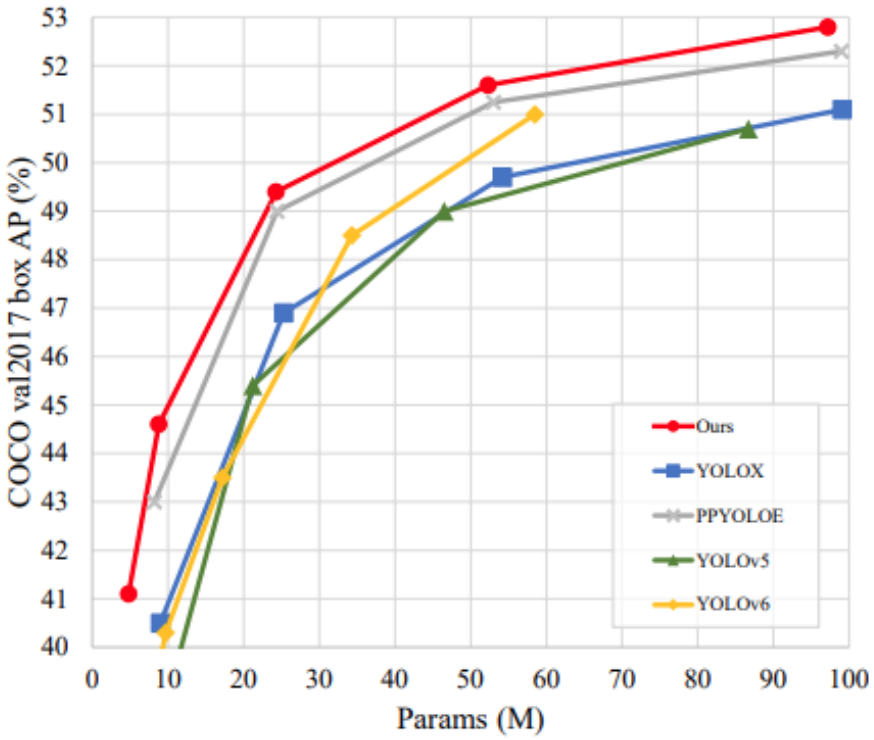


***Figure 3.4.*** *Comparison of parameters and accuracy of RTMDet and other state-of-the-art real-time object detectors on the COCO val2017 dataset. From [46].*

RTMDet uses a dynamic soft label assignment strategy in training that is based on the SimOTA label assignment. The typical way to handle label assignment in training, is to choose the prediction that has the highest IoU value with the ground truth (and the IoU value crosses a certain threshold), and then assign it to that ground truth [12]. According to SimOTA authors, this way of labeling without context can be sub-optimal, because it ignores the different sizes, shapes and occlusion conditions of different objects [12, 13]. SimOTA is introduced in the YOLOX paper, and it uses the global context of the image to assign the labels, by calculating the cost between all predictions for each ground truth, and then assigning the positive anchors that had the lowest cost value by using a dynamic top-k algorithm [13]. RTMDet changes the cost calculations in multiple ways, but the most significant one is introducing soft labels to calculating the classification cost. Soft labels mean that the labels have a probability value from 0 to 1 instead of hard labels like binary

labels, that can cause a prediction with a wrong bounding box, but a high classification score to still get a low classification cost score [46].

The paper reports state of the art performance on both the COCO dataset, with the basic RTMDet method, and on the DOTA dataset, with the modified version of the method, that is designed for oriented object detection, named RTMDet-R. RTMDet does not offer anything new for oriented object detection specifically, because when going from basic RTMDet to RTMDet-R, the only changes are adding the angle parameter as the fifth parameter, decoding OBBs instead of HBBs and changing the loss function to one that is suitable for oriented bounding boxes [46].

The MMRotate implementation uses the "CSPNeXt" backbone that is pretrained with the ImageNet-1K dataset. The RTMDet configs only have the le90 bounding box definition version available.

## 3.5 PP-YOLOE-R

Baidu/PaddlePaddle have released their own family of YOLO based detectors, of which PP-YOLOE-R is the latest one [70]. The first iteration of these PaddlePaddle detectors was published in 2020, and it was called PP-YOLO. PP-YOLO is heavily based on YOLOv3 and tries to improve on it's precision using existing tricks, while maintaining similiar amount of parameters [44]. The tricks take some inspiration from YOLOv4 and include things like removing the grid sensitivity problem, where it is harder to predict the bounding box center points from boxes that are closer to the edges of grids, and using a more optimized NMS-function [44, 3]. Unlike YOLOv4, it does not use the CSP-based backbone, but still manages to outperform YOLOv4 on the COCO dataset [44].

Subsequent PaddlePaddle methods are PP-YOLOv2 (2021) and PP-YOLOE (2022) [25, 75]. PP-YOLOE takes inspiration from the YOLOX method, which unlike the original YOLO series, is an anchor-free method. The anchor-freeness in PP-YOLOE is achieved by copying the FCOS method and using every pixel in the image as an potential anchor point. For every ground truth object in the image, the centering pixels in the feature map are selected as positive samples in training, this requires an additional hyper-parameter to determine how large is the central portion of a ground truth bounding box [63, 75]. Then in the inference process, the detector tries to find center points of objects and regress the bounding box around it.

Like the newer RTMDet method, PP-YOLOE also loans the dynamic label assignment style of YOLOX, but more specifically it uses the task alignment learning (TAL), that is proposed in the paper of an object detection method called TOOD [11]. Like SimOTA, TAL dynamically assigns positive anchor points for each ground truth, but it does this separately for labels and bounding boxes and then uses this information to find the best

of both worlds. This combination of classification and localization tasks allows the finding of the anchor point that has the most precise bounding box and also the highest classification score that is possible. Most conventional single-stage detectors have a head each in parallel for classification and localization, PP-YOLOE again follows TOOD, and proposed an Efficient Task-aligned Head (ET-head), which improves the interaction between the classification and localization tasks, preventing misalignment when detecting densely placed objects [75, 11]. For the backbone and neck, PP-YOLOE combines the residual connections of ResNet and the dense connections of CSPNet and proposes the RepResBlock network, which uses CSP-blocks and PAN feature extractor. According to the PP-YOLOE paper, it slightly beats both YOLOv5 and YOLOX on the COCO test-dev dataset [75].

PP-YOLOE-R was also published in 2022, and it focuses on changes, that enable PP-YOLOE to perform oriented object detection, with good performance, and minimal parameter and computation costs. The four main changes are:

- PP-YOLOE-R uses a new loss for bounding box regression called ProbIoU, which uses Gaussian distribution representations of the OBBs to compare the similarity between objects. This loss function was created with oriented object detection in mind, and it helps with the boundary discontinuity problem. [70, 34]
- TAL is changed into Rotated Task Alignment Learning, by just simply changing IoU to SkewIoU, when calculating the task alignment metric, that is used to select the positive samples with [70]. SkewIoU takes into account the rotated nature of the bounding boxes.
- For predicting the fifth angle parameter, an additional decoupled head is added. Usually oriented object detectors regress all five parameters in the same head, but the authors assume that predicting the angle parameter requires different features to learn. [70]
- When calculating ProbIoU, ellipse-like Gaussian bounding boxes (GBBs) are created from the OBBs. This causes a difficulty when trying to figure out the orientation of the GBB when the OBB is roughly square shaped. This problem is solved by using Distribution Focal Loss (DFL) to predict the angle. DFL models the angles as General distributions to provide the angle predictions [32, 70].

In the PaddleDetection implementation there are four different sizes of CSPRepResNet to choose from as the backbone. The dataset that CSPRepResNet is pretrained with is not documented anywhere, but it can be assumed to be ImageNet1K, from some comments in the PaddleDetection GitHub Issues. Only the OpenCV option for oriented bounding box definition is available.

## 3.6 YOLOv5 + Circular Smooth Label

YOLOv5 is a bit unique in a way, because Ultralytics has not released any research paper for it explaining the architecture and improvements on YOLOv4 in detail. But there is a documentation with an architecture summary, and the code for it is available, if one wants to do their own research. YOLOv4 is released on the C based Darknet framework and YOLOv5 is released on the Python based PyTorch framework [3, 65].

YOLOv5 utlizes the CSP-blocks and uses the CSPDarknet53 as YOLOv4 does [65, 3]. Spatial Pyramid Pooling (SPP) is a technique used in object detection architectures, that allows the use of variable image input sizes, without cropping or warping the input images [21]. SPP is a layer that comes after the backbone network's last layer, at the start of the neck, and it produces a fixed sized output, no matter the input size [21]. SPP is used in YOLOv4, but it is replaced in YOLOv5, with a reportedly faster version called Spatial Pyramid Pooling Fusion (SPPF), where it is also used to output different scale representations of the input feature maps [65]. After that, the neck features a PAN to replace FPN in gathering the backbone's different scale outputs for the prediction phase, similarly to YOLOv4 [3, 65]. The head part of the network is the same as it has been since YOLOv3 [57, 3, 65].

YOLOv4 featured and introduced a multitude of data augmentation techniques and training strategies, which YOLOv5 also uses. When training object detectors, data augmentation methods are often used to alter the dataset images, in an effort to diversify the training data and subsequently improving the the final detectors performance. YOLOv5 offers these augmentations: Mosaic, Copy-Paste, Random Affine Transformations, MixUp, Albumentations library, HSV and random horizontal/vertical flips [65].

Mosaic Augmentation is an newer method, since it was introduced in the YOLOv4 paper. It takes four images from the dataset and combines them into a single image, without the images overlapping with each other. This combining aims to help the network to learn different scales of objects located in different parts of the image [65]. Kaixuan Hu is the implementer of the OBB using YOLOv5, which is used in this thesis. In his blog, Hu speculates that Mosaic Augmentation has the greatest effect in the betterment of the detector and reports over a 7 percentage-point mAP increase, when going from not using Mosaic Augmentation to using it, training YOLOv5 on the DOTA dataset [23].

YOLOv5 also provides advanced training strategies like AutoAnchor, which calculates optimal anchor boxes to use based on the ground truth boxes of your dataset [65]. This removes the need to manually adjust the amount, scale and aspect ratio of anchor boxes to optimize for a dataset.

Hu combines YOLOv5 with the use of Circular Smooth Labels(CSL), which changes the angle prediction from a regression problem into a classification problem, by having 180

different classes, one for each degree [78]. This removes the boundary discontinuity problem. The visualization of the adjacency of the labels can be seen in Figure 3.5.

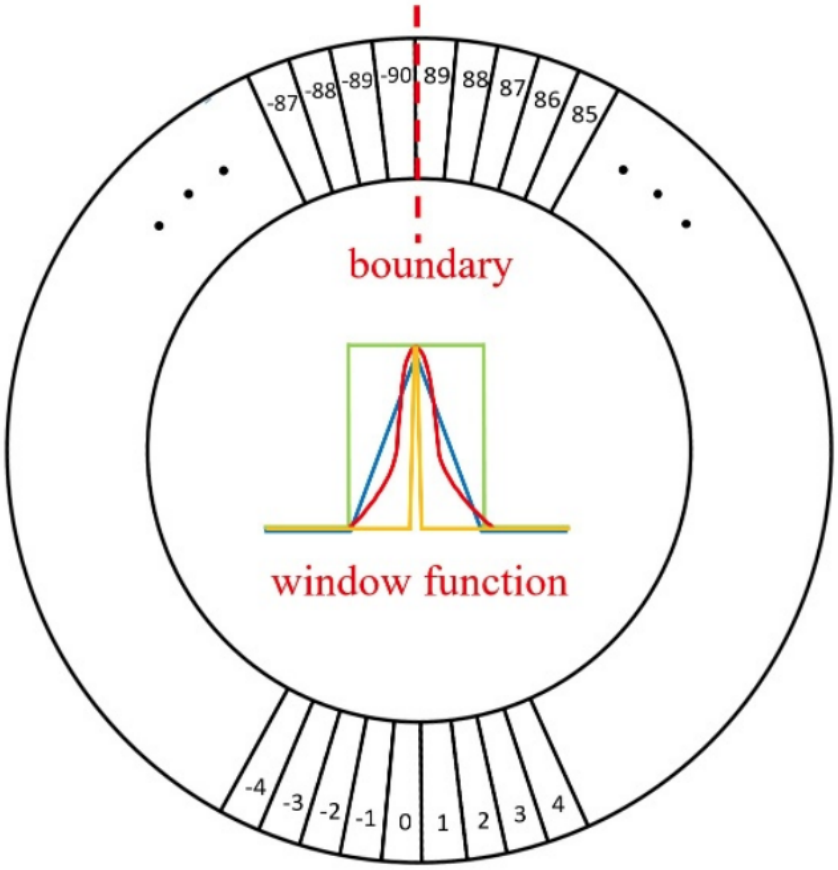


***Figure 3.5.*** *Representation of Circle Smooth Labeling. From [78].*

By encoding the classes in a circular manner, the loss value of predictions -89° and 89° with regards to ground truth value -90° is the same, as it ought to be [78, 23]. The CSL paper features 4 window functions of which the Gaussian was measured to be the best [78]. Hu changed the radius value of CSL's window function from the default 6 to 2 for increased performance on DOTA [23].

Hu Kaixuan has implemented the required steps to make YOLOv5 use oriented bounding boxes using code from the official GitHub implementations of CSL, Oriented R-CNN and DOTA. The bounding box definition is the 90° Long Edge definition, as it is also recommended in the CSL paper [78, 23].

# 4. DATASET AND EXPERIMENTS

In this chapter the created dataset is presented, along with an outline of its creation methodology and adequacy for the task. After that the training and evaluation pipelines are explained, followed by the presentation and analysis of the evaluation results on the created dataset.

## 4.1 Created Dataset

To find out the most suitable oriented object detection method for the surveillance point of view use case, a dataset called Tampere Parking Lot(TPL) was created by aiming a Basler BIP2-1600c camera towards a parking lot from the third floor window of a building. A total of 820 images of size 1920x1080 were captured using a 400 second interval between captures to achieve alternating weather and lighting conditions in the images. Figure 4.1 shows an example of this from TPL with 3 images.

***Figure 4.1.*** *Example images from the dataset (with license plates blurred)*

In the annotation process, the only class defined was the VEHICLE class, which encompasses all possible vehicles that appear in the images: cars, vans and trucks. The oriented bounding boxes for the vehicles were annotated manually using an open source annotation tool originally developed by Intel, called Computer Vision Annotation Tool(CVAT). It can be seen from Figure 4.2 that the overlapping problem of HBBs is not as significant in this created dataset as it can be in aerial images, like mentioned in Chapter 2, or simply when the parking spaces are not as axis-aligned to the camera as they are in the TPL dataset.

Figure 4.2 does not actually have actual OBB rectangles with 90° angles on the left side, because the bounding boxes are manually annotated by hand. In the training pipeline

for this dataset, the annotations are run through an algorithm that creates minimum area rectangles from the annotated bounding boxes to create the OBBs that are used to train the network. These actual final ground-truth annotations can be seen in Figure 4.3, which also demonstrates that OBBs can also have problems representing objects, because of the extra background, which is present in the VEHICLE bounding box annotations that are in the middle of the image.

***Figure 4.2.*** *OBB-like polygon annotations on the left and HBB annotations on the right.*

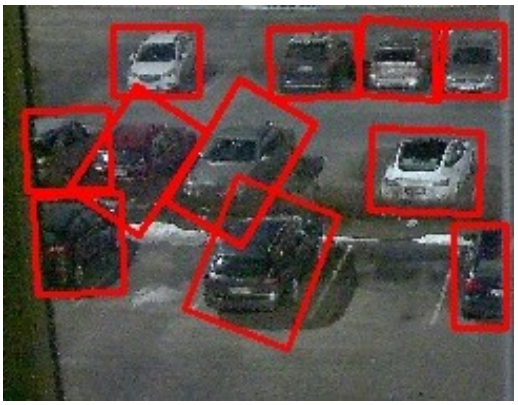

***Figure 4.3.*** *Actual ground-truth annotations, that are given to networks when training/evaluating*

Figure 4.4 shows an example on how HBBs are not fit for detecting trailers on a truck parking area, but the OBBs are able to handle it, especially when singling out only the roof of the trailers. The images also demonstrate how OBBs are not the perfect representation for the roofs in this defined perspective, but they would be in a true aerial image taken with a satellite/drone. This is caused by lowering the perspective, which transforms these objects from rectangular to quadrilateral, as the angles change. So at least in this way, this defined bird's eye view can be more complex for orinted bounding box detection compared to aerial images. One advantage of the defined bird's eye view is that no aerial technology, like drones or satellites are needed for gathering data.

The TPL dataset was divided into a training set of 494 images and a test set of 326 images for evaluation. Evaluation was performed using COCO API's COCOEval class,

which calculates the precision for ten threshold values from 0.05 to 0.95. After that we take the average value from those and end up with the final mAP value.

***Figure 4.4.*** *OBB annotations on the left are able to single out the trailers while the HBBs on the right can not.*

## 4.2 Evaluation and results

The objective of the evaluations featured in this chapter is to find the best performing method on the created TPL dataset. Another goal was to use the same evaluation pipeline, including framework and pre- and post-processing programming scripts, to offer fair comparison of the methods in the evaluation metrics of precision (mAP) and speed (FPS). For all the training and evaluation runs, the used CPU is Intel Core i9-7920X and the GPU is NVIDIA GeForce RTX 3080.

Figure 4.5 shows two example images of the evaluation detection OBBs with YOLOv5+CSL and the ground-truth values. On the left image, two missed cars can be seen, and on the right image, an extra false positive detection is present. The detected OBBs can also have very different angles, compared to the more oriented ground-truth bounding boxes, which can be explained by the minimal visual difference of the cars between the less oriented ground-truth boxes and the more oriented ground-truth boxes. This causes the orientation of the detected OBBs to be somwehere in between.

Training the detectors was done with the default settings of each method. The dataset images were rescaled and the input size was set to 512x512 for the final evaluations. Annotations were downloaded from CVAT and converted to the DOTA format, which has support from all of the methods. The format looks like: "x1, y1, x2, y2, x3, y3, x4, y4, category, difficult" [8], where "category" in this case is the VEHICLE class. The "difficult" parameter can be used to seperate the difficulty of annotations from each other by setting a float number from 0-1, to indicate the rising difficulty of detecting this object. In this case it was set to 0 in all annotations, and none of the methods use it in their training.

In training, an epoch is a complete pass through all of the training dataset images, after which the model's weights are updated and often the current model is evaluated. The YOLOv5 implementation has a patience parameter for stopping the training and creation

***Figure 4.5.*** *Ground-truth annotations in red and YOLOv5+CSL detections in green*

of new epochs, when the evaluations during the training phase have not improved in precision for a certain number of epochs, resulting in a model that has gone through just the right amount of passes through the dataset to give out the best possible score. PaddleDetection also has an –eval flag in the training script to calculate the best performing epoch, while training a set amount of epochs. To guarantee that all methods have the best possible final evaluation result, a patience script was also created for MMRotate methods.

NMS IoU threshold was set to 0.1 for all methods, to ensure that NMS would not be the reason for evaluation result differences. For YOLOv5, minimal data augmentations were set with no use of Mosaic augmentation, 0.1 chance of MixUp augmentation, No horizontal or vertical flips, HSV of (0.015, 0.7, 0.4) and no Copy-Paste augmentation. For Random Affine Transformations: random scaling between 0.8-2, translation of 0.1 and rotation of 180 degrees. Otherwise all learning rates and momentums and other parameters were left at their default values for all methods. All MMRotate models use horizontal/vertical/diagonal Random flip data augmentation with a 0.75 probability. RTMDet-R and PP-YOLOE-R also use random rotation with a 0.5 probability.

Initially OpenVINO was planned to be used as the model framework for all the evaluations, until MMRotate/MMDeploy had showed to have no support for it. The problem in converting the MMRotate ONNX models to OpenVINO was the MMRotate models' NMS algorithm, which is integrated to the network code and only had TensorRT support. Another problem with the integrated MMRotate NMS is that there is no apparent way to disable it from the configurations. In an effort to help the conversion from ONNX to OpenVINO to work, the MMRotate source code for the Rotated RetinaNet method was modified by removing the NMS algorithm, which lead to a successfull conversion, but abnormally slow inference speeds. This led to the choosing of TensorRT being the model framework for all evaluations, which also meant that all MMRotate methods would use

a different NMS implementation for their evaluations, slightly decreasing the validity of inference speed comparisons as NMS was included in the inference speed calculations.

For some methods MMRotate offers three different bounding box definitions in their configurations: The OpenCV definition(oc), the 135° Long Edge definition(le135) and the 90° Long Edge definition(le90). All of these were tested with Rotated RetinaNet and RoI Transformer methods in Table 4.1. along with two HBB options in Rotated RetinaNet, Multi-scale Random rotation(MS RR) and floating-point 16(fp16) versions. These Rotated RetinaNet detectors were trained with 1024x1024 sized images, all the rest of the detectors featured in this chapter were trained with 512x512 scaled images to allow for fair comparison. Using smaller resolution input images makes the detector faster, but resizing the input images can also make objects harder to detect, worsening the precision. The HBB versions performed slightly worse compared to the OBB versions, which is good to see, as there was doubt if the TPL dataset is too simple to benefit from oriented bounding boxes. The data augmentations of MS RR seemed to have no positive effect compared to the general le90 configurations in neither speed nor precision. Fp16 uses less computer memory and to no surprise, is not an improvement compared to the le90 version, except for the minuscule FPS difference with Rotated RetinaNet. But the FPS differences are basically non-existent, except for surprising drop in both precision and speed of the RoI Transformer OpenCV version. The reason for the drop is left unknown, and it might be a problem with the MMRotate implementation of RoI Transformer+oc. The OBB 90° Long Edge performed the best overall and was chosen to be the definition to be used in all remaining MMRotate evaluations. Gliding Vertex and RTMDet-R also only has the le90 configuration available, which supports the choice. In PaddleDetection, PP-YOLOE-R only has the OpenCV definition available, but the YOLOv5 implementation also uses the 90° Long Edge definition.

Some of the used methods have different sized backbone configurations, which affect the amount of layers in the network. Smaller ones for when FPS is the most important factor, and large ones for best possible mAP. These methods are RTMDet-R, PP-YOLOE-R and YOLOv5 and all of them have four or five different configuration choices from smallest to largest. With the smallest YOLOv5 version YOLOv5s, an mAP of 0.405 and FPS of 112.4 was achieved. This result was set as a reference point for the other methods, and this is why Table 4.2. which features the evaluation results of RTMDet-R only has small and medium configurations tested, as YOLOv5s results already outperform RTMDet-R medium in both mAP and FPS, which suggests that evaluating the large version could not be better than YOLOv5, as the inference times are guaranteed to rise, when the network size is increased. Table 4.2. also features the Multi-sampling configuration results, which interestingly sligthly increased mAP and greatly reduced FPS for the small version of RTMDet-R, but no such effect can be seen with the medium version.

Table 4.3. features all the PP-YOLOE-R evaluations. From the OpenVINO results, it can

***Table 4.1.*** *Evaluations of Rotated RetinaNet and RoI Transformer on the TPL dataset*

| Method | Bounding box | mAP | FPS |
|---|---|---|---|
| Rotated RetinaNet (1024x1024) OpenVINO | HBB le90 | 0.299 | 2.19 |
| | HBB le135 | 0.263 | 2.22 |
| | OBB le90 | **0.318** | 2.22 |
| | OBB le135 | 0.291 | 2.19 |
| | OBB(MS RR) le90 | 0.305 | 2.17 |
| | OBB oc | **0.318** | 2.17 |
| | OBB(fp16) le90 | 0.276 | **2.24** |
| RoI Transformer (512x512) TensorRT | OBB le90 | **0.361** | 41.84 |
| | OBB le135 | 0.292 | **42.2** |
| | OBB(MS RR) le90 | 0.317 | 41.2 |
| | OBB oc | 0.112 | 18.8 |
| | OBB(fp16) le90 | 0.358 | 40.3 |

***Table 4.2.*** *RTMDet-R evaluations on the TPL dataset*

| Method | Size | mAP | FPS |
|---|---|---|---|
| RTMDet-R TensorRT | s | 0.250 | **103.1** |
| | m | **0.353** | 40.3 |
| | s (MS) | 0.284 | 55.6 |
| | m (MS) | 0.339 | 40.2 |

be seen that the large version had the highest mAP. The reason why x is less precise compared to l is most probably because the training dataset is so small and the network has so much capacity that it is overfitting, meaning that it adjusts too precisely to the training dataset and as a result, performs worse on unseen data. At first glance it appeared that the Multi-sampling versions should not have any effect, since the setting only seems to change the used directory from dataset/dota/ to dataset/dota_ms/, which are not used anyway, since custom data is being used. But looking at the results, it seemed to have a slight negative effect on the mAP of every version of the method.

Table 4.4. shows the final best results for every method used. YOLOv5 + CSL takes the cake and eats it too by having the highest mAP of 0.405 and sharing the highest FPS of 112.4 with PP-YOLOE-R. RoI Transformer and RTMDet both have around 0.05 less mAP compared to YOLOv5, making them the 2nd and 3rd most accurate detectors in this evaluation, respectively. Gliding Vertex and PP-YOLOE-R have around a 0.10 mAP gap to the best detector, making them the 4th and 5th most accurate detectors, respectively. The least accurate detector is Rotated RetinaNet with a mAP that is 0.165 points worse

***Table 4.3.*** *PP-YOLOE-R evaluations on the TPL dataset*

| Method | Size | mAP | FPS |
|---|---|---|---|
| PP-YOLOE-R OpenVINO | s | 0.220 | 35.0 |
| | m | 0.277 | 25.6 |
| | l | **0.299** | 17.1 |
| | x | 0.278 | 10.0 |
| | s (MS) | 0.216 | **35.7** |
| | m (MS) | 0.255 | 25.5 |
| | l (MS) | 0.213 | 17.4 |
| | x (MS) | 0.225 | 10.1 |
| PP-YOLOE-R TensorRT | m | 0.278 | **147.1** |
| | l | **0.299** | 112.4 |

than YOLOv5.

From the YOLO based methods of RTMDet-R PP-YOLOE-R and YOLOv5 + CSL, the only anchor-based detector, which is YOLOv5, was the best, which is not surprising, considering the simplicity of objects in the dataset, making it so that an anchor-based solution would have no hard time finding objects. Another thing to consider is that both RTMDet-R and PP-YOLOE-R implementations are part of large object detection toolboxes, so their implementation may be less optimized and thought out compared to the single implementation of YOLOv5 + CSL by Kaixuan Hu.

RoI Transformer has an impressive performance coming 2nd best in accuracy, while being released in 2018, this might also be explainable by the semi-horizontal alignment of the objects in the TPL dataset, as RoI Transformer firstly generates the HBB proposals from the images and then transforms them to OBBs afterwards, so the RRoI Learner need not learn large offsets when rotating the bounding boxes. RetinaNet is the oldest method of the bunch and except for the Focal Loss, it mostly resembles the basic Faster R-CNN method. So it is not surprising, that Rotated RetinaNet holds lasts place in terms of accuracy.

FPS-wise there is a clear division into two groups: MMRotate methods and others. It is surprising to see that the MMRotate methods do not differ in inference speed almost at all, considering that they do have very different ways of solving the detection problem. All of them except RTMDet-R have the ResNet-50 as the backbone, which might mostly explain the similarity. RTMDet-R is also the one of the four methods that deviates most from the median FPS of all the MMRotate methods.

Now the most interesting part of the FPS results is the big difference between the MMRotate methods and other methods. One clear possible reason is the different NMS imple-

***Table 4.4.*** *Final best evaluation results for each method on the TPL dataset*

| **Method** | **mAP** | **FPS** |
|---|---|---|
| Rotated RetinaNet | 0.240 | 44.2 |
| Gliding Vertex | 0.307 | 44.2 |
| RoI Transformer | 0.361 | 41.8 |
| RTMDet-R | 0.353 | 40.3 |
| PP-YOLOE-R | 0.299 | **112.4** |
| YOLOv5 + CSL | **0.405** | **112.4** |

mentation, which all of the MMRotate detectors have integrated in the head of their models. The second possible reason is that all MMRotate models could have some shared computational overhead, which happens in the creation/deployment of all MMRotate models which is not optimal and slows things down. The third suspect was the fact that the MMRotate methods have an extra data conversion in the evaluation pipeline, because the inference output was different compared to other methods. MMRotate methods outputs the five parameter representation $x, y, w, h, \theta$, which is converted to the eight parameter bounding box representation, to allow the rest of the pipeline to be exactly identical across all methods. This conversion time was calculated in the pipeline and the median result was, that it takes about 0.32 milliseconds. Removing 0.32 ms from a 40 FPS detector would result in a 40.52 FPS detector, so the output data conversion has very insignificant impact.

# 5. CONCLUSIONS

This thesis explored oriented bounding box detection, with a focus on bird's eye view scene imagery, by researching relevant literature and testing out the detection accuracy and speed of six different oriented object detection methods, of which five were implemented by popular open source computer vision toolbox creators and one was an individual GitHub user adding oriented bounding box detection to a popular open source object detection method. The tested methods were:

- Rotated RetinaNet,
- RoI Transformer,
- Gliding Vertex and
- RTMDet-R, from the MMRotate oriented object detection toolbox.
- PP-YOLOE-R, from the PaddleDetection object detection toolbox.
- YOLOv5 + CSL, using the Ultralytics YOLOv5 modified by Kaixuan Hu to use oriented bounding boxes and Circular Smooth Labels.

For the purpose of testing the methods, a dataset with oriented bounding box "VEHICLE" class annotations was created with images collected from a bird's eye view camera aimed at a local parking lot. The selected oriented object detection methods were trained and evaluated on separate training- and test sets. The methods were evaluated using the most common accuracy and speed metrics used in oriented object detection.

**Which open-source oriented bounding box detection method performs the best on bird's eye view images?** YOLOv5 + CSL was proven to be the best method on all metrics out of all the methods, with mAP of 0.405 and FPS of 112.4, making it the fastest and most accurate method on the created dataset, and thus it can be recommended to be used in similar use cases, regardless of whether real-time detection is needed or not.

**Does the bird's eye view object detection benefit from using oriented bounding boxes?** Yes, the usefulness of oriented bounding box detection methods over generic horizontal bounding box detection was tested with Rotated RetinaNet, where using OBBs instead of HBBs on the created dataset slightly(1.9 percentage-points) increased the accuracy. The three most popular OBB definitions were also tested with Rotated RetinaNet

***Table 5.1.*** *The detectors ranked by accuracy on the created dataset, while presenting their architectural attributes.*

| Method | Anchor-based | Angle-based | Single-stage |
|---|---|---|---|
| 1. YOLOv5 + CSL | X | X | X |
| 2. RoI Transformer | X | X | |
| 3. RTMDet-R | | X | X |
| 4. Gliding Vertex | X | | |
| 5. PP-YOLOE-R | | X | X |
| 6. Rotated RetinaNet | X | X | X |

and RoI Transformer, resulting in the 90° Long Edge definition being slightly more accurate compared to the others.

**Are there some OBB detection method key attributes that are always better than the others?** Table 5.1 features the evaluated methods in a descending accuracy order, and their architectural attributes. The table demonstrates that no meaningful correlations could be found in the accuracy order of the evaluations in regard to if the method were anchor-based, angle-based or single-stage or not. Thus no general recommendations can be made based on those properties, although there has not been much enthusiasm around two-stage detector research in the recent years. The best performing method YOLOv5+CSL is anchor-based, angle-based and single-stage. All the three different YOLO-based models were also ranked very differently on the table.

Another observation was, that the MMRotate methods all had a similar inference speed, which was significantly slower compared to the other methods (around 40 FPS). This slowness could maybe be caused by some computation overhead coming from the deployment of the MMRotate detection models using MMDeploy. This is a reason to not recommend MMRotate methods on real-time detection problems, unless a fix for the speed is found.

The created dataset, as said in chapter 4, did not have that difficult objects for a HBB detector to detect, as the parking lot was very axis-aligned to the camera, so the vehicles were mostly horizontally or vertically positioned in the images. The created dataset was also pretty small, which usually means there is less variability in the data.

For further development a new dataset could be created, with objects/annotations that would be more rotated, and thus would put more emphasis on the detectors' ability to detect oriented bounding boxes. And a larger accuracy difference might be seen between detectors using HBBs and OBBs in training. The amount of images in the new dataset should also be larger, while maintaining good diversity, to be able to see which methods are best able to learn complex patterns and relationships and generalize to unseen data.

If there is more interest in the use case of fully top down aerial imagery, like satellite images, then the DOTA-v2.0 dataset, with its 11,268 images, is a great choice to use as a benchmark [8].

Out of all the used methods in this thesis, the YOLOv5 would of course be the most interesting to research further as it already performed the best out of all of them. With YOLOv5, Ultralytics offers about 30 hyperparameters, which can be modified and would allow for a lot of testing [65]. The more interesting parameters would be the different data augmentation techniques like mosaic augmentation, which was not used at all in this thesis. YOLOv5 also provides three larger model sizes: m,l and x, which could presumably offer better accuracy with less inference speed.

Ultralytics has also released their newest version of YOLO called YOLOv8 in 2023. Compared to YOLOv5, YOLOv8 has some architectural differences, like the fact that it is an anchor-free detection method. This is done by cutting up the image into grid cells and predicting the center of an object withing the cell where the object is. Ultralytics also supports oriented bounding box training/detecting, and DOTA-v1.0 pretrained models for YOLOv8, which makes testing the method out simpler. [28]

# REFERENCES


[1] Zakaryia Almahasees and Sameh Mahmoud. “Evaluation of Google Image Translate in Rendering Arabic Signage into English”. In: *World Journal of English Language* 12 (Feb. 2022), pp. 185–197. DOI: 10.5430/wjel.v12n1p185.

[2] PaddlePaddle Authors. *PaddleDetection, Object detection and instance segmentation toolkit based on PaddlePaddle.* https://github.com/PaddlePaddle/PaddleDetection. 2019.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[4] Gong Cheng et al. “Anchor-Free Oriented Proposal Generator for Object Detection”. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–11. ISSN: 1558-0644. DOI: 10.1109/tgrs.2022.3183022. URL: http://dx.doi.org/10.1109/TGRS.2022.3183022.

[5] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[6] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[7] Jian Ding et al. “Learning RoI Transformer for Oriented Object Detection in Aerial Images”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2844–2853. DOI: 10.1109/CVPR.2019.00296.

[8] Jian Ding et al. “Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.11 (Nov. 2022), pp. 7778–7796. DOI: 10.1109/tpami.2021.3117983. URL: https://doi.org/10.1109%2Ftpami.2021.3117983.

[9] Kaiwen Duan et al. *CenterNet: Keypoint Triplets for Object Detection*. 2019. arXiv: 1904.08189 [cs.CV].

[10] Mohamed Elgendy. *Deep Learning for Vision Systems*. eng. 1st ed. New York: Manning Publications Co. LLC, 2020. ISBN: 1617296198.

[11] Chengjian Feng et al. *TOOD: Task-aligned One-stage Object Detection*. 2021. arXiv: 2108.07755 [cs.CV].

[12] Zheng Ge et al. *OTA: Optimal Transport Assignment for Object Detection*. 2021. arXiv: 2103.14259 [cs.CV].

[13] Zheng Ge et al. *YOLOX: Exceeding YOLO Series in 2021*. 2021. arXiv: 2107.08430 [cs.CV].

[14] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083.

[15] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV].

[16] Jack Greenhalgh and Majid Mirmehdi. "Recognizing Text-Based Traffic Signs". In: *IEEE Transactions on Intelligent Transportation Systems* 16 (June 2015), pp. 1–10. DOI: 10.1109/TITS.2014.2363167.

[17] Jiaming Han et al. "Align Deep Features for Oriented Object Detection". In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–11. DOI: 10.1109/TGRS.2021.3062048.

[18] Jiaming Han et al. *ReDet: A Rotation-equivariant Detector for Aerial Object Detection*. 2021. arXiv: 2103.07733 [cs.CV].

[19] Jing Han et al. "Multi-Oriented and Scale-Invariant License Plate Detection Based on Convolutional Neural Networks". In: *Sensors* 19.5 (2019). ISSN: 1424-8220. DOI: 10.3390/s19051175. URL: https://www.mdpi.com/1424-8220/19/5/1175.

[20] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[21] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 346–361. ISBN: 9783319105789. DOI: 10.1007/978-3-319-10578-9_23. URL: http://dx.doi.org/10.1007/978-3-319-10578-9_23.

[22] Xu He et al. "Learning Rotated Inscribed Ellipse for Oriented Object Detection in Remote Sensing Images". In: *Remote Sensing* 13.18 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13183622. URL: https://www.mdpi.com/2072-4292/13/18/3622.

[23] Kaixuan Hu. *Modify the trap record of YOLOv5 rotating target by yourself (translated)*. https://www.zhihu.com/column/c_1358464959123390464. Accessed: 7.1.2024.

[24] Kaixuan Hu. *yolov5_obb*. https://github.com/hukaixuan19970627/yolov5_obb. 2021.

[25] Xin Huang et al. *PP-YOLOv2: A Practical Object Detector*. 2021. arXiv: 2104.10419 [cs.CV].

[26] Yingying Jiang et al. "R2 CNN: Rotational Region CNN for Arbitrarily-Oriented Scene Text Detection". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018, pp. 3610–3615. DOI: 10.1109/ICPR.2018.8545598.

[27] Glenn Jocher. *Ultralytics YOLOv5*. Version 7.0. 2020. DOI: 10.5281/zenodo.3908559. URL: https://github.com/ultralytics/yolov5.

[28] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLOv8*. Version 8.0.0. 2023. URL: https://github.com/ultralytics/ultralytics.

[29] Dimosthenis Karatzas et al. “ICDAR 2015 competition on Robust Reading”. In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. 2015, pp. 1156–1160. DOI: 10.1109/ICDAR.2015.7333942.

[30] Hei Law and Jia Deng. *CornerNet: Detecting Objects as Paired Keypoints*. 2019. arXiv: 1808.01244 [cs.CV].

[31] Wentong Li et al. *Oriented RepPoints for Aerial Object Detection*. 2022. arXiv: 2105.11111 [cs.CV].

[32] Xiang Li et al. *Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection*. 2020. arXiv: 2006.04388 [cs.CV].

[33] Zeming Li et al. *Light-Head R-CNN: In Defense of Two-Stage Object Detector*. 2017. arXiv: 1711.07264 [cs.CV].

[34] Zhonghua Li et al. *FCOSR: A Simple Anchor-free Rotated Detector for Aerial Object Detection*. 2021. arXiv: 2111.10780 [cs.CV].

[35] Minghui Liao, Baoguang Shi, and Xiang Bai. “TextBoxes++: A Single-Shot Oriented Scene Text Detector”. In: *IEEE Transactions on Image Processing* 27.8 (Aug. 2018), pp. 3676–3690. DOI: 10.1109/tip.2018.2825107. URL: https://doi.org/10.1109%2Ftip.2018.2825107.

[36] Minghui Liao et al. *Rotation-Sensitive Regression for Oriented Scene Text Detection*. 2018. arXiv: 1803.05265 [cs.CV].

[37] Tsung-Yi Lin et al. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV].

[38] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[39] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[40] Shu Liu et al. *Path Aggregation Network for Instance Segmentation*. 2018. arXiv: 1803.01534 [cs.CV].

[41] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 21–37. ISBN: 9783319464480. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[42] Xuebo Liu et al. *FOTS: Fast Oriented Text Spotting with a Unified Network*. 2018. arXiv: 1801.01671 [cs.CV].

[43] Jeffri M. Llerena et al. *Gaussian Bounding Boxes and Probabilistic Intersection-over-Union for Object Detection*. 2021. arXiv: 2106.06072 [cs.CV].

[44] Xiang Long et al. *PP-YOLO: An Effective and Efficient Implementation of Object Detector*. 2020. arXiv: 2007.12099 [cs.CV].

[45] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-

1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[46] Chengqi Lyu et al. *RTMDet: An Empirical Study of Designing Real-Time Object Detectors*. 2022. arXiv: 2212.07784 [cs.CV].

[47] Jianqi Ma et al. "Arbitrary-Oriented Scene Text Detection via Rotation Proposals". In: *IEEE Transactions on Multimedia* 20.11 (2018), pp. 3111–3122. DOI: 10.1109/TMM.2018.2818020.

[48] Umberto. Michelucci. *Advanced Applied Deep Learning Convolutional Neural Networks and Object Detection*. 1st ed. 2019. Berkeley, CA: Apress, 2019.

[49] Qi Ming et al. "Optimization for Arbitrary-Oriented Object Detection via Representation Invariance Loss". In: *IEEE Geoscience and Remote Sensing Letters* 19 (2022), pp. 1–5. DOI: 10.1109/lgrs.2021.3115110. URL: https://doi.org/10.1109%2Flgrs.2021.3115110.

[50] Roohie Naaz Mir et al. *Advancement of Deep Learning and its Applications in Object Detection and Recognition*. 1st ed. Milton: River Publishers, 2023.

[51] Eoghan Mulcahy, Pepijn Van de Ven, and John Nelson. "Aerial Object Detection for Water-Based Search & Rescue". In: *Artificial Intelligence and Cognitive Science*. Ed. by Luca Longo and Ruairi O'Reilly. Cham: Springer Nature Switzerland, 2023, pp. 344–354. ISBN: 978-3-031-26438-2.

[52] *OpenCV documentation*. https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html. Accessed: 2024-01-26.

[53] Wen Qian et al. *Learning Modulated Loss for Rotated Object Detection*. 2019. arXiv: 1911.08299 [cs.CV].

[54] Yuhao Qing et al. "Improved YOLO Network for Free-Angle Remote Sensing Target Detection". In: *Remote Sensing* 13.11 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13112171. URL: https://www.mdpi.com/2072-4292/13/11/2171.

[55] "Real-time object detection in agricultural/remote environments using the multiple-expert colour feature extreme learning machine (MEC-ELM)". eng. In: *Computers in industry* 98 (2018), pp. 183–191. ISSN: 0166-3615.

[56] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242 [cs.CV].

[57] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[58] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[59] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[60] Rajalingappaa Shanmugamani. *Deep learning for computer vision : expert techniques to train advanced neural networks using TensorFlow and Keras*. 1st edition. Birmingham, England: Paths International Ltd, 2018.

[61] Wenzhong Shi et al. “Airborne LiDAR for Detection and Characterization of Urban Objects and Traffic Dynamics”. eng. In: *Urban Informatics*. Singapore: Springer Singapore Pte. Limited, 2021. ISBN: 9811589828.

[62] Jung-il Shin et al. “Using UAV Multispectral Images for Classification of Forest Burn Severity—A Case Study of the 2019 Gangneung Forest Fire”. eng. In: *Forests* 10.11 (2019), pp. 1025–. ISSN: 1999-4907.

[63] Zhi Tian et al. *FCOS: Fully Convolutional One-Stage Object Detection*. 2019. arXiv: 1904.01355 [cs.CV].

[64] J. R. R. Uijlings et al. “Selective Search for Object Recognition”. In: *International Journal of Computer Vision* 104.2 (2013), pp. 154–171. URL: https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013.

[65] Ultralytics. *YOLOv5: A state-of-the-art real-time object detection system*. https://docs.ultralytics.com. Accessed: 19.12.2023. 2021.

[66] Andreas Veit et al. *COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images*. 2016. arXiv: 1601.07140 [cs.CV].

[67] P. Viola and M. Jones. “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*. Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

[68] Chien-Yao Wang et al. *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. 2019. arXiv: 1911.11929 [cs.CV].

[69] Jian Wang, Le Yang, and Fan Li. “Predicting Arbitrary-Oriented Objects as Points in Remote Sensing Images”. In: *Remote Sensing* 13.18 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13183731. URL: https://www.mdpi.com/2072-4292/13/18/3731.

[70] Xinxin Wang et al. *PP-YOLOE-R: An Efficient Anchor-Free Rotated Object Detector*. 2022. arXiv: 2211.02386 [cs.CV].

[71] Haoran Wei et al. *Oriented Objects as pairs of Middle Lines*. 2020. arXiv: 1912.10694 [cs.CV].

[72] Gui-Song Xia et al. *DOTA: A Large-scale Dataset for Object Detection in Aerial Images*. 2019. arXiv: 1711.10398 [cs.CV].

[73] Zhifeng Xiao et al. “Axis Learning for Orientated Objects Detection in Aerial Images”. In: *Remote Sensing* 12.6 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12060908. URL: https://www.mdpi.com/2072-4292/12/6/908.

[74] Xingxing Xie et al. *Oriented R-CNN for Object Detection*. 2021. arXiv: 2108.05699 [cs.CV].

[75] Shangliang Xu et al. *PP-YOLOE: An evolved version of YOLO*. 2022. arXiv: 2203.16250 [cs.CV].

[76] Yongchao Xu et al. “Gliding Vertex on the Horizontal Bounding Box for Multi-Oriented Object Detection”. In: *IEEE Transactions on Pattern Analysis and Machine*

*Intelligence* 43.4 (Apr. 2021), pp. 1452–1459. DOI: 10.1109/tpami.2020.2974745. URL: https://doi.org/10.1109%2Ftpami.2020.2974745.

[77] Xue Yang and Junchi Yan. "Arbitrary-Oriented Object Detection with Circular Smooth Label". In: *CoRR* abs/2003.05597 (2020). arXiv: 2003.05597. URL: https://arxiv.org/abs/2003.05597.

[78] Xue Yang and Junchi Yan. *On the Arbitrary-Oriented Object Detection: Classification based Approaches Revisited*. 2022. arXiv: 2003.05597 [cs.CV].

[79] Xue Yang et al. *Dense Label Encoding for Boundary Discontinuity Free Rotation Detection*. 2021. arXiv: 2011.09670 [cs.CV].

[80] Xue Yang et al. *Learning High-Precision Bounding Box for Rotated Object Detection via Kullback-Leibler Divergence*. 2022. arXiv: 2106.01883 [cs.CV].

[81] Xue Yang et al. *R3Det: Refined Single-Stage Detector with Feature Refinement for Rotating Object*. 2020. arXiv: 1908.05612 [cs.CV].

[82] Xue Yang et al. *Rethinking Rotated Object Detection with Gaussian Wasserstein Distance Loss*. 2022. arXiv: 2101.11952 [cs.CV].

[83] Xue Yang et al. *SCRDet: Towards More Robust Detection for Small, Cluttered and Rotated Objects*. 2019. arXiv: 1811.07126 [cs.CV].

[84] Xue Yang et al. *SCRDet++: Detecting Small, Cluttered and Rotated Objects via Instance-Level Feature Denoising and Rotation Loss Smoothing*. 2022. arXiv: 2004.13316 [cs.CV].

[85] Xue Yang et al. *The KFIoU Loss for Rotated Object Detection*. 2023. arXiv: 2201.12558 [cs.CV].

[86] Ze Yang et al. *RepPoints: Point Set Representation for Object Detection*. 2019. arXiv: 1904.11490 [cs.CV].

[87] Yanqing Yao et al. "On Improving Bounding Box Representations for Oriented Object Detection". In: *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), pp. 1–11. DOI: 10.1109/TGRS.2022.3231340.

[88] Donghang Yu et al. "An Anchor-Free and Angle-Free Detector for Oriented Object Detection Using Bounding Box Projection". In: *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), pp. 1–17. DOI: 10.1109/TGRS.2023.3305729.

[89] Zenghui Zhang et al. "Toward Arbitrary-Oriented Ship Detection With Rotated Region Proposal and Discrimination Networks". In: *IEEE Geoscience and Remote Sensing Letters* 15.11 (2018), pp. 1745–1749. DOI: 10.1109/LGRS.2018.2856921.

[90] Qiang Zhou et al. *Point RCNN: An Angle-Free Framework for Rotated Object Detection*. 2022. arXiv: 2205.14328 [cs.CV].

[91] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. *Bottom-up Object Detection by Grouping Extreme and Center Points*. 2019. arXiv: 1901.08043 [cs.CV].

[92] Xinyu Zhou et al. *EAST: An Efficient and Accurate Scene Text Detector*. 2017. arXiv: 1704.03155 [cs.CV].

[93] Yue Zhou et al. “MMRotate: A Rotated Object Detection Benchmark using PyTorch”. In: *Proceedings of the 30th ACM International Conference on Multimedia*. ACM, Oct. 2022. DOI: 10.1145/3503161.3548541. URL: http://dx.doi.org/10.1145/3503161.3548541.