UNIVERSIDADE FEDERAL FLUMINENSE

IGOR GARCIA BALLHAUSEN SAMPAIO

A systematic approach for object detection using deep learning and CAD models

NITERÓI

2020

UNIVERSIDADE FEDERAL FLUMINENSE

IGOR GARCIA BALLHAUSEN SAMPAIO

A systematic approach for object detection using deep learning and CAD models

Dissertation presented to the Graduate School of Computer Science of Universidade Federal Fluminense as a partial requirement for obtaining the Master of Science degree in Computer Science. Field: Systems and Information Engineering

Advisor:
JOSÉ VITERBO FILHO

Co-Advisor:
JORIS MICHEL GÉRARD DANIEL GUÉRIN

NITERÓI

2020

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S192s Sampaio, Igor Garcia Ballhausen
A systematic approach for object detection using deep learning and CAD models / Igor Garcia Ballhausen Sampaio ; José Viterbo Filho, orientador ; Joris Michel Gérard Daniel Guérin, coorientador. Niterói, 2020.
82 f.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2020.

DOI: http://dx.doi.org/10.22409/PGC.2020.m.10464898790

1. Detecção de objetos. 2. Visão computacional. 3. Aprendizado profundo. 4. Rede neural convolucional. 5. Produção intelectual. I. Filho, José Viterbo, orientador. II. Guérin, Joris Michel Gérard Daniel, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecário responsável: Sandra Lopes Coelho - CRB7/3389

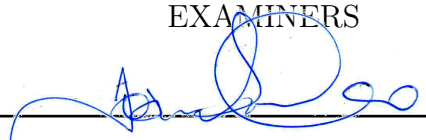
Igor Garcia Ballhausen Sampaio

A systematic approach for object detection using deep learning and CAD models

Dissertation presented to the Graduate School of Computer Science of Universidade Federal Fluminense as a partial requirement for obtaining the Master of Science degree in Computer Science. Field: Systems and Information Engineering.

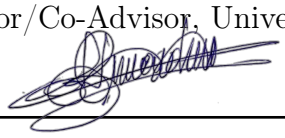
Approved in December, 2020.

EXAMINERS



Prof. Dr. José Viterbo Filho - Orientador/Advisor, UFF

Dr. Joris M. G. Daniel Guérin - Coorientador/Co-Advisor, Université de Toulouse



Prof. Dr. Flávia Cristina Bernardini, UFF

Prof. Dr. Esteban Walter Gonzalez Clua, UFF

Prof. Dr. Fátima de L. S Nunes Marques, USP

Niterói

2020

"The good thing about science is that it's true whether or not you believe in it."

Neil deGrasse Tyson

Acknowledgements

To my family members who have supported me so far, I want them to know that I recognize the incentive, the trust and all the bases that made me who I am today.

My mother, who was always my greatest supporter and never missed anything so I could finish my studies and reach my goals.

My wife, for having walked beside me, for her patience, understanding, especially for always presenting a smile, when I sacrificed the days, nights, weekends and holidays in order to carry out this study.

To this institution, I am grateful for the favorable environment for evolution and growth, as well as for all the people who make it so special for those who know it.

To my advisors, I recognize the deep trust, wisdom, availability and opportunity. Always giving me resources and tools to evolve a little more every day.

To my friends I leave a word of gratitude for all the support and inspiration.

I would like to thank all the people who interfered in my journey, because in some way they influenced it.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Resumo

A Detecção de Objetos (OD) é um importante problema de visão computacional para a indústria, que pode ser usado para controle de qualidade nas linhas de produção. Recentemente, métodos de aprendizado profundo possibilitaram o treinamento de modelos OD com desempenho muito bom para conjuntos de dados complexos do mundo real. No entanto, a adoção desses modelos na indústria foi limitada pela dificuldade - e pelo custo significativo - de coletar conjuntos de dados de treinamento de alta qualidade. Por outro lado, OD para controle de qualidade em linhas de produção apresenta a especificidade de ter modelos CAD disponíveis para os objetos a serem detectados. Neste artigo, apresentamos um método totalmente automatizado que usa um modelo CAD de um objeto e retorna um modelo OD totalmente treinado para detectar esse objeto. Para fazer isso, criamos um script do Blender que gera conjuntos de dados realistas de imagens contendo o modelo CAD e os rótulos OD correspondentes. Esses conjuntos de dados são usados para treinar os detectores de objetos. O método é validado experimentalmente em um exemplo prático, e mostramos que essa abordagem pode gerar modelos OD com bom desempenho, 98.27% de precisão no melhor caso, em imagens reais, enquanto é treinado apenas em imagens sintéticas.

Palavras-chave: Detecção de objetos, modelos CAD, geração de imagens sintéticas, visão computacional, aprendizado profundo, rede neural convolucional.

Abstract

Object Detection (OD) is an important research topic in computer vision. It can be used in industrial scenarios for quality control in production lines. Recently, deep learning methods have enabled the training of models for OD with very good performance for complex real world datasets. However, the adoption of these models in industry has been limited by the difficulty — and the significant cost — of collecting high quality training datasets. On the other hand, OD for quality control in production lines presents the specificity of often having CAD models available for the objects to be detected. In this paper, we introduce a fully automated method that uses a CAD model of an object and returns a fully trained OD model for detecting this object. To do this, we created a Blender script that generates realistic datasets of images containing the CAD model as well as the corresponding OD labels. These datasets are then used for training the object detectors. The method is validated experimentally on a practical example, and we show that this approach can generate OD models performing well, 98.27 % precision in the best case on real images, while being trained only on synthetic images.

Keywords: Object Detection, CAD Models, Synthetic image generation, Computer Vision, Deep Learning, Convolutional Neural Network.

List of Figures

List of Tables

List of Acronyms

API	:	Application Programming Interface;
AUC	:	Area under the curve;
AI	:	Artificial Intelligence;
AP	:	Average Precision;
CNN	:	Convolutional Neural Network;
CAD	:	Computer-aided Design;
CPS	:	Cyber Physical Systems;
DCNN	:	Deep Convolutional Neural Network;
DPM	:	Deformable Part-based Model;
GPU	:	Graphic Power Unit;
HOG	:	Histogram of Oriented Gradients;
IoT	:	Internet of Things;
IoU	:	Intersection over Union;
MPU	:	Digital Mock-up;
mAP	:	Mean Average Precision;
OD	:	Object Detection;
RCNN	:	Region-based Convolutional Neural Networks;
SVM	:	Support Vector Machine;

Contents

Chapter 1

Introduction

Object Detection (OD) is an important research topic in computer vision that has been widely studied over the last decade [70, 73, 32, 67, 53]. It consists in finding the coordinates of bounding boxes encapsulating objects of interest in an image [43]. It has applications in robotics systems, industrial process control systems and tracking systems, among other application domains [11]. As example, object detection and pose estimation is frequently the first step of robotic systems. Recently, deep learning methods, such as those employing Convolutional Neural Networks (CNNs), have become the standard tool for object detection, outperforming alternatives in object recognition benchmarks [37]. As for industrial process control systems, the task of detecting objects in images for industrial process control systems is particularly challenging and of great interest in the context of Industry 4.0. The major steps of such systems are the observation of the production context and analysis of the observed images for the extraction of the labels, which will correspond to the classification of these images [55]. And finally, real-time object detection and tracking have shown to be the basis of intelligent production for industrial 4.0 applications. It is a challenging task because of various distorted data in complex industrial setting [34]. In Figure 1.1 below, we can see an example of object detection in a vehicle production line.

A CNN differs from standard Artificial Neural Network (Fully Connected) in the structure of its hidden layers, which enables it to have a significantly higher number of neurons per layer. CNN architectures have the advantage of automatically learning filters for extracting features, thus avoiding the complicated and time consuming feature engineering process (thresholding, contouring, segmentation, clustering). Also the expressivity and robust training algorithms allow to learn informative object representations without the need to design features manually [72]. Many different types of CNNs have been developed

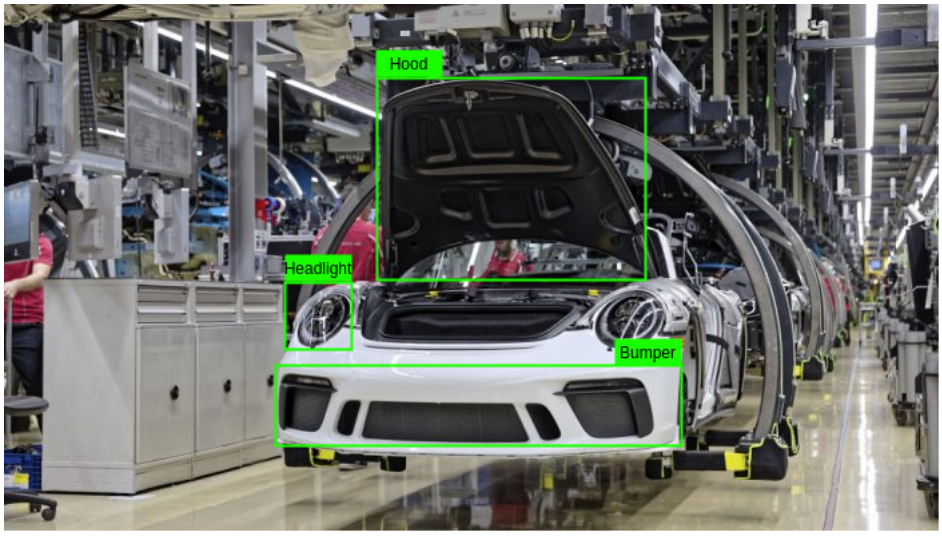


Figure 1.1: Example of object detection on an industrial production line.

with different hidden layer structures. Some examples include Alexnet [26], MobileNet [20] and Inception [59].

One of the typical issues with deep learning is the requirement of large labelled datasets for training. Although there are various online databases containing millions of images, for specific industrial applications, it can be necessary to create custom datasets containing the objects of interest. Most object detection datasets represent scenarios from either everyday life or mobile robotic environments. While this scenario is important from both a research and application standpoint, it was found that industrial applications, such as bin picking or surface and defect inspection, have quite different characteristics that are not modeled by the existing datasets [20]. This includes different 3D shapes, different kinds of sensors and modalities, and different kinds of object placements. As a result, methods that perform well on existing datasets sometimes show quite different results when applied to industrial scenarios without retraining [11]. The process of generating a specific dataset for retraining is tedious, and can be error-prone when conducted by non-professional technicians. Moreover, generating a new image dataset and labelling it manually can be very time consuming and expansive [21].

On the other hand, the specific case of OD for industrial production lines presents the specificity that the manufacturers often have access to the CAD models of the objects to detect. Thanks to advances in computer graphics techniques, such as ray tracing [54], the generation of photo-realistic images has become possible. In such artificially generated images, the computer can be employed for bounding box labelling. As deep neural networks are typically trained on Graphical Processing Units (GPUs), these can

also be used to render large quantities of images efficiently [21]. Besides that, the use of synthetic images rendered from CAD models to train OD models has already been proposed in [41], as well as in other works, such as [44, 17], for example. However, their approaches requires manual scene creation by Blender artists, it is not automated, and the objects used are usually generic, such as buses, airplanes, cars, animals, etc.

Identifying the appropriate objects and separating them from the faulty ones, is the main goal of automatic object detection system. Thus, the use of automation in any industrial process increases the investment cost initially. However, it generates greater efficiency in manufacturing, reducing operating costs and causing fewer errors and less man-machine intervention. Automation in this sector means using control systems to handle different processes and machines to replace human efforts [56].

1.1 Problem Definition

The automation of the detection and identification of objects in the industrial process is crucial and may have great impact on the overall productivity of the industry. Therefore, the first step to obtain this type of automation would be the definition of the process for the generation of object classification models, which will allow the identification of flaws in the production line. The generation of these models requires the aggregation of large datasets of images of objects in different situations.

However, collecting real images can be extremely laborious. In [17], for example, each real image was collected in different contexts - disordered background, changes in lighting, etc. In addition, it must be ensured that each object is shown from different poses. For this reason, the use of CAD models has already been used as an alternative to this problem [17, 57, 60, 11]. However, there is no study that says what the standard of synthetic image generation in the industry should be, in order to guarantee an efficient result. Working with synthetic images does not guarantee an efficient result if there is no systematization of the process of generating these synthetic images with the necessary variations. So, this work proposes a systematic methodology for generating synthetic images using our variation pattern.

In addition, good results are typically obtained by training CNNs using datasets that involve a very large number of labeled images, as in the case of ImageNet [26]. For this reason, in this work, in addition to the attempt to systematize the generation of labeled images, the training of these CNN models was done with a very small number of images,

saving time.

1.2 Objectives

The main objective of this work is to propose a systematic approach to detect and identify objects in the industrial scenario. Hence, we could devise three specific objectives of this work: (i) to develop a systematic process for generating labeled synthetic images, (ii) to train a convolutional neural network model with a reduced number of images, in order to save time, and finally, (iii) to compare the performance of the models using real, synthetic and hybrids datasets. Therefore, this work can be divided into three parts: automated image generation from properly labeled CAD objects, training a convolutional neural network model, with specific adjustments, in order to achieve a good performance with a reduced number of images to detect objects on the assembly line, and finally, to evaluate the efficiency of this proposed approach by comparing the object detection performance using synthetic, hybrid and real images datasets. In addition, the dataset generated during this work, as well as the complete systematic approach were made publicly available so that other users can test and improve it, if applicable.

1.3 Methodology

Firstly, a study of the current literature was made with regard to object detection, use of CAD models to generate synthetic image datasets and the metrics commonly used to evaluate CNN models for object detection.

After studying the literature, we defined the process for the automatic generation of realistic labeled images containing the objects to be detected from CAD models. In sequence, a study was carried out to demonstrate that the images can be used to adjust the DO models and that they can work well in real images of the object. Subsequently, we carried out a set of experiments to study the influence of each parameter on the final results and to identify the ideal configuration for our approach. Finally, we validated our approach using the CAD model of some objects for training, as well as labeled real images containing the real object for evaluation.

1.4 Contributions

This master's thesis contributes with a new scalable algorithm for the generation of synthetic images from CAD models and a systematic approach that aims to establish parameters and performance standards in the results. Therefore, we can separate these contributions into two parts: technological and scientific.

From this, in the technological contribution item, we can list the construction of a new algorithm, in the scientific item, we can list the new systematic approach proposed in this work, in addition to the experiments, discussions and results about the use of hybrid datasets (synthetic images and real) in the scope of object detection using CAD models.

1.5 Organization

The structure of this work is divided into six chapters. In Chapter 2, the general concept about object detection, automation of industrial processes, anomaly detection in assembly lines, computer vision, industry 4.0, deep learning applied to image recognition and evaluation metrics is presented, describing the main approaches and methods used. Then, Chapter 3 presents a review of the literature on the different works carried out regarding the detection of objects with real and synthetic images, as well as the contexts, applications and results obtained, making a qualitative and quantitative comparison with this work. In Chapter 4, the systhematic approach developed in this work is presented, as well as the generation of synthetic images, the training of neural networks and hyperparameters tuning. Chapter 5 describes the experiments, the datasets used, the test parameters, the results and the hypothesis tests comparing the use of real, synthetic and hybrid images datasets. Finally, chapter 6 presents the conclusions of this work, describing some limitations of the research and the future work.

Chapter 2

Fundamental Concepts

This chapter describes the fundamental concepts in the relationship between industry 4.0, visual computing, object detection and deep learning algorithms. In addition, several techniques and methods are described within the scope of this relationship. Finally, the metrics used for the evaluation of object detection models are presented and discussed, so that their importance in the analysis of the results of this work is clear.

2.1 Industry 4.0

Industry 4.0 represents the current trend in automation technologies in the manufacturing sector and includes mainly enabling technologies, such as cyber-physical systems (CPS), Internet of Things (IoT) and cloud computing. This concept represents the technological evolution from embedded systems to cyber-physical systems. In Industry 4.0, embedded systems, semantic machine-to-machine communication, IoT and CPS technologies are integrating virtual space with the physical world; in addition, a new generation of industrial systems, such as smart factories, is emerging to deal with the complexity of production in the cyber-physical environment [68].

Under this hypothesis, the term industry 4.0 (or fourth industrial revolution) would not only imply a technological change, but versatile organizational implications as well. As a result, a change from product- to service-orientation is sought and expected: manufacturing and service industry will become complementary, encouraging a new form of production [10].

According to [4], the Industry 4.0 has nine pillars of technology, including virtual reality, artificial intelligence, industrial internet, industrial big data, industrial robot, 3D

printing, cloud computing, knowledge work automation and industrial network security. And at least 4 of these pillars are used and joined in this work (artificial intelligence, industrial big data, industrial robot and knowledge work automation).

2.2 Computer Vision and Object Recognition

Computer Vision and Pattern Recognition study a lot of problems, like image matching, clustering and segmentation, extraction of invariant interest points, feature selection, optimal classifier design, model selection and many others. The computational analysis of images and more abstract patterns is a complex and challenging task, which demands interdisciplinary efforts [49].

However, over the last years deep learning methods have been shown to outperform previous state-of-the-art machine learning techniques in several fields, with computer vision being one of the most prominent cases. Various computer vision tasks, such as object detection, face recognition, action and activity recognition and human pose estimation use machine learning techniques [65].

Object recognition is one of the fundamental challenges in computer vision, which generally consists of two different types of tasks: object instance recognition and object class recognition. The first type aims at identifying previously seen object instances such as a specific car, and is largely a matching problem in which the differences between the stored exemplars and the objects to be reidentified in an input image are mainly caused by imaging condition changes, and hence can be effectively handled by some alignment process. The second type, also known as category-level or generic object recognition, focuses on recognizing (always unseen-before) instances of some predefined categories [70]. In addition, it supports a wide range of practical applications, such as tracking and surveillance [35], anomaly detection [51], augmented reality [45], segmentation and pose estimation [40], for example.

Basically, an object detection (OD) system can be easily described according to Figure 2.1, which shows the basic steps that are involved in the OD process [53]. The basic entry for the OD system can be an image or a scene, in the case of videos. The basic objective of this system is to detect objects that are present in the image/scene or simply, in other words, the system needs to categorize and localize the various objects in the respective object classes.

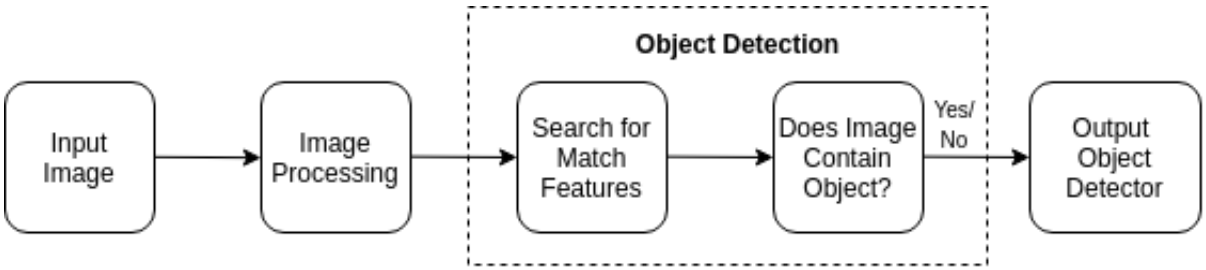


Figure 2.1: Object detection model. Adapted from [53]

So, the first step in the object detection process is to input the image into the object detector / classifier system. After that, this set of images will be processed (feature extraction, resize map, etc.). Soon after, comes the object detection module itself, that is, the search for match features and compare them with the object in question, in order to check if the object is contained therein. Finally, the object detector is created.

A system based on object detection can be defined as a data labeling and localizing problem, based on known object models. From an image, which contains one or more objects of interest in a given set of labels corresponding to a set of models known to the system, it is expected that the system assigns correct labels to the regions of the image.

Therefore, the OD system based on deep learning consists of three main phases: the learning phase, the validation phase and the inference phase, which are shown in Figure 2.2 - which shows the operation of the OD system. The learning phase consists of training the apprentice, so that he recognizes the objects present in the image, which is provided as input to the system. In the model validation phase, the trained model is evaluated in order to provide its ability to detect the object in question. In the inference phase, the classifier makes use of learning, which was done in the learning phase, to classify objects. With that, after the image processing, the direct correspondence of the model is made, producing the characteristics of an object in the image. The main objective of this phase is to locate the object and decide whether that same object is present in the image, which is supplied to the system as an input and, if so, to which class of object it belongs.

Unraveling the three phases involved in object detection, we obtain two phases in common: image processing and feature extraction. Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Image processing basically includes the following two steps [23]:

- Analyzing and manipulating the images which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.

- Output is the last stage in which result can be altered image or report that is based on image analysis.

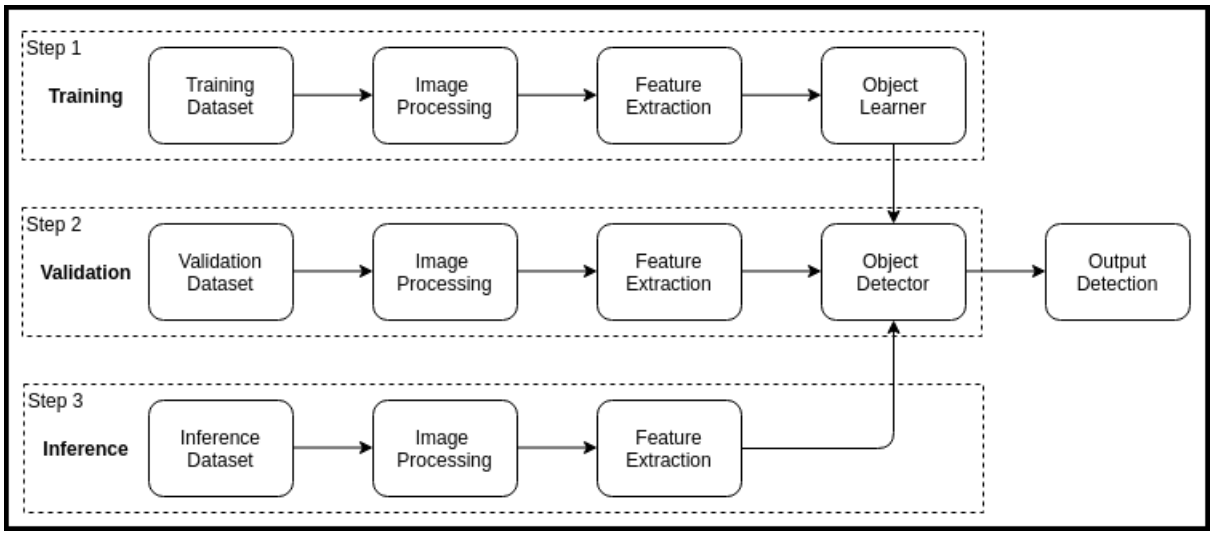


Figure 2.2: Operation of the object detection model.

Feature extraction, on the other hand, is used to extract the most distinct characteristics present in a dataset (in our case - images) that are used to represent and describe the data. Thus, fundamentally uses features within the object to identify, locate and process regions of interest [52]. In [61], for example, it is possible to note a comparative study of the various combinations of representations of image features, such as the region-based, global and local block-based categorization of the image database.

Finally, the object learner phase, which consists of the detection learning method, has several approaches, with several different methods. According to [73], which makes a history review on object detection, the first unrestricted method for object detection (in this case with human faces) was with [63, 64], where the author follows a most straight forward way of detection, ie, sliding windows. After that, detectors based on Histogram of Oriented Gradients (HOG) feature descriptor was proposed [7]. Soon after, the Deformable Part-based Model (DPM) [14] appeared. Also according to [73], the performance of hand-crafted features became saturated, object detection has reached a plateau after 2010. However, with the emergence of deep neural networks, capable of learning representations of robust and high characteristics level of an image, a lot of networks have

been proposed for object detection. Since then, object detection started to evolve at an unprecedented speed.

It is important to note that nowadays, the part of image processing and the extraction of features are already included in CNNs, as in [47, 26], for example. Therefore, the model is simplified, as shown in Figure 2.3.

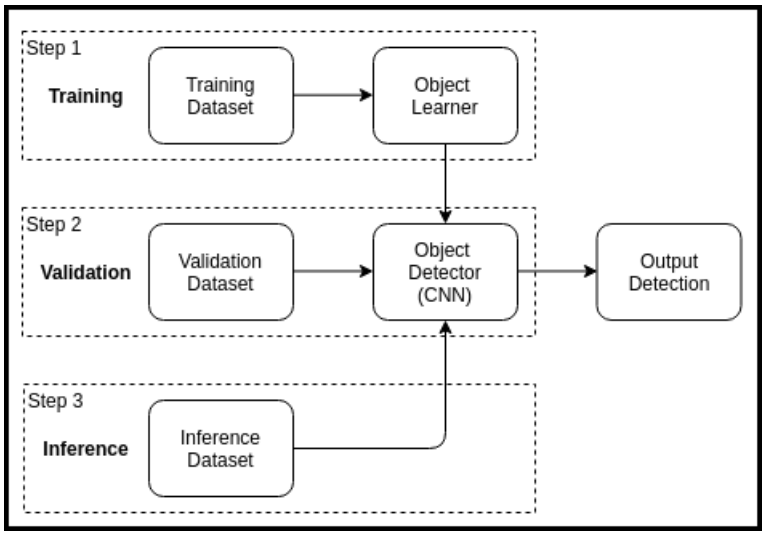


Figure 2.3: Operation of the object detection model with CNN.

2.3 Object Detection with Deep Learning

As stated above and according to [32], deep learning techniques have emerged as a powerful strategy for learning characteristic representations directly from data and have led to significant advances in the field of generic object detection.

With all this progress of deep neural networks applied to the object detection context, different network architectures have been developed, mainly convolutional neural networks (CNN): Regions with CNN (RCNN) [62], Spatial Pyramid Pooling (SSPNet) [16], Fast RCNN [15], Faster RCNN [47], Feature Pyramid Networks (FPN) [28], You Only Look Once (YOLO) [46], Single Shot MultiBox Detector (SSD) [33], RetinaNet [29] and many others have been proposed.

2.3.1 Convolutional Neural Networks

The Convolutional Neural Networks (CNN) are very similar to fully-connected Neural Networks. They are made up of neurons that have learnable weights and biases. Each

neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. They still have a loss function on the last (fully-connected) layer and all the tips/tricks has developed for learning regular Neural Networks still apply [24].

With the revival of convolutional neural networks in [26] and their application to object detection problems, we can see in Figure 2.4 the general architecture of a CNN.

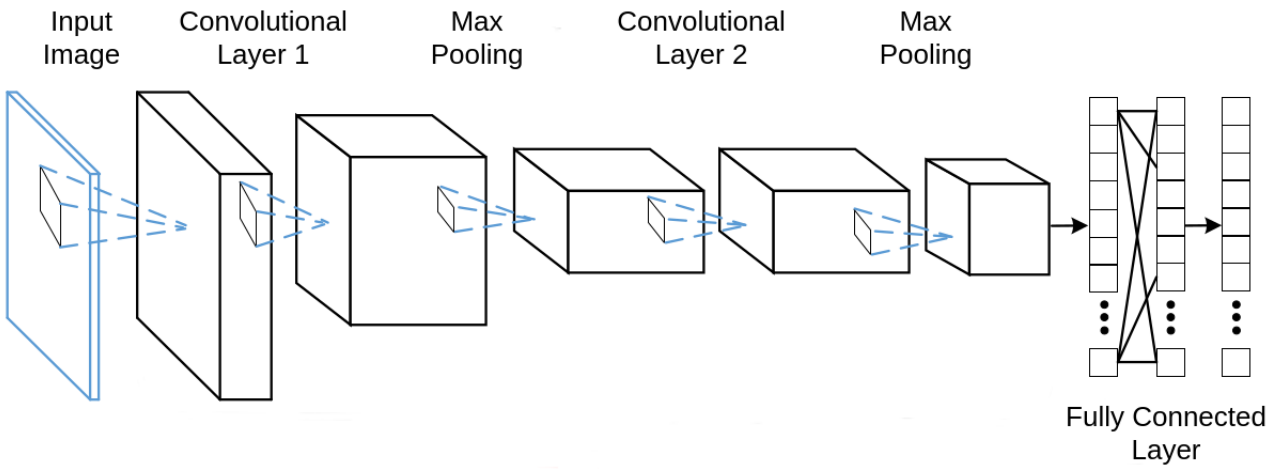
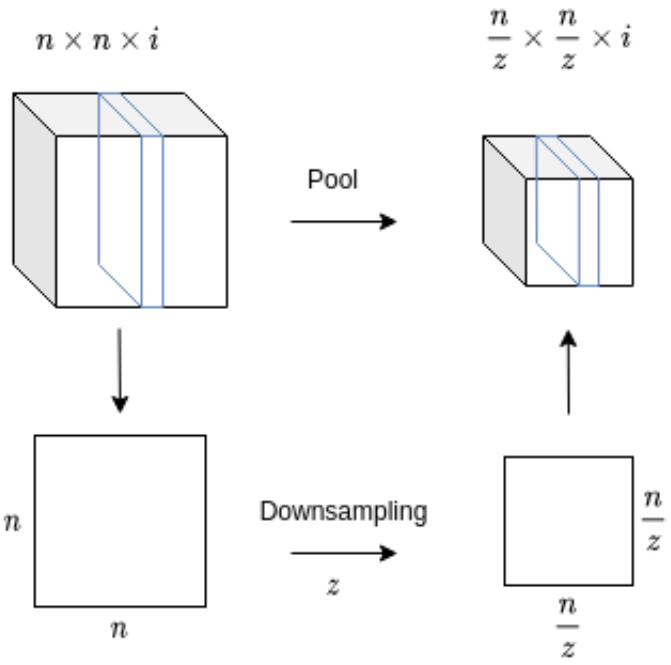


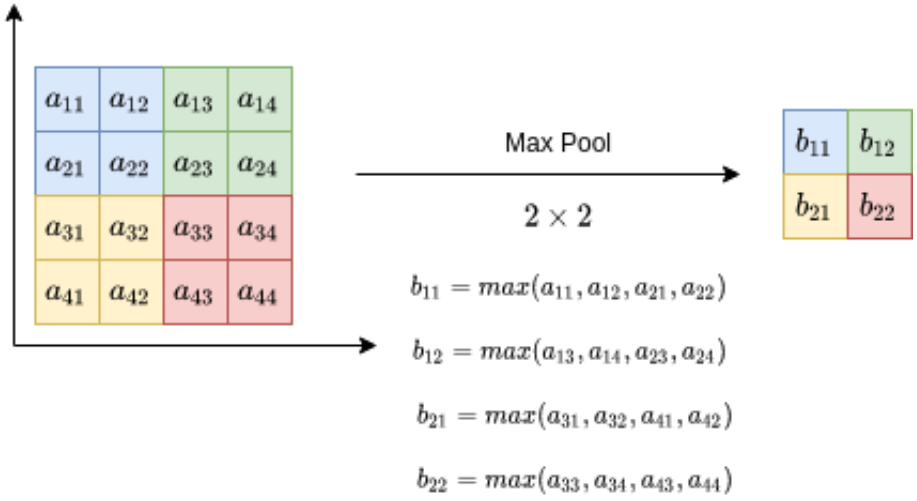
Figure 2.4: Architecture of general CNN. Adopted from [24]

The CNN consists of multiple layers. Each layer takes a multi-dimensional array of numbers as input and produces another multi-dimensional array of numbers as output (which then becomes the input of the next layer). When classifying images, the input to the first layer is the input image ($n \times n$), while the output of the final layer is a set of likelihoods of the different categories. A simple CNN is a sequence of layers, and every layer of a CNN transforms one volume of activations to another through a differentiable function as we can see in [71].

Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. In Figure 2.5(a), the input size $[n \times n \times i]$ is pooled with filter size z, producing an output size $[\frac{n}{z} \times \frac{n}{z} \times i]$, where n is the image resolution and i is the number of samples. Notice that the volume depth is preserved. The most common downsampling operation is max [24], giving rise to max pooling, as example shown in Figure 2.5(b).



(a) Downsampling



(b) Max pooling operation example with 2x2 window

Figure 2.5: CNN scheme for downsampling and max pooling.

2.3.2 Faster R-CNN

According to [47], recent advances in object detection are driven by the success of region proposal methods and region-based convolutional neural networks (R-CNNs). Although region-based CNNs were computationally expensive, their cost has been drastically reduced. The latest incarnation, Fast R-CNN, achieves near real-time rates using very

deep networks, when ignoring the time spent on region proposals. Now, proposals are the computational bottleneck in state-of-the-art detection systems. A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score.

Faster R-CNN, the model used in this work, takes as input an entire image and a set of object proposals. The network first processes the whole image with several convolutional and max pooling layers to produce a convolutional feature map. Then, for each object proposal, a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map, as seen in Figure 2.6.

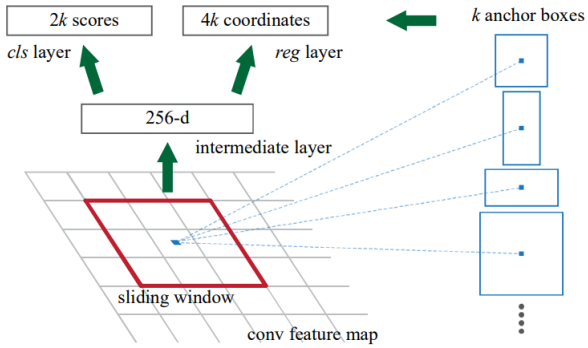


Figure 2.6: Region Proposal Network. Adapted from [47]

Each feature vector is fed into a sequence of fully connected layers that finally branch into two sibling output layers: one that produces softmax probability estimates over K object classes plus a catch-all "background" class and another layer that outputs four real-valued numbers for each of the K object classes. Each set of 4 values encodes refined bounding-box positions for one of the K classes [15].

2.4 Evaluation Metrics for Object Detection

The purpose of this section is to summarize some common evaluation metrics of the object detection problem. In object detection, a number of well-known datasets and benchmarks have been released in the past years, including the datasets of PASCAL VOC Challenges [1] [13, 12], ImageNet Large Scale Visual Recognition Challenge [2] [50], MS-COCO Detection Challenge [3] [30], Open Images Detection (OID) challenge [4] [25].

[1] http://host.robots.ox.ac.uk/pascal/VOC/
[2] http://image-net.org/challenges/LSVRC/
[3] http://cocodataset.org/
[4] https://storage.googleapis.com/openimages/web/index.html

However, before presenting the metrics used in competitions, we must first establish two fundamental metrics.

- **Confidence score:** is the probability that an anchor box contains an object. It is usually predicted by a classifier.
- **Intersection over Union (IoU):** is defined as the area of the intersection divided by the area of the union of a predicted bounding box (A) and a ground-truth box (B) as the equation 2.1 shows:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

Therefore, Intersection over Union, also known as Jaccard index [22], is used to determine the accuracy of an object detector on a specific dataset. This metric is often used for object detection challenges, such as the popular PASCAL VOC challenge [13, 12]. This metric is typically used to assess the performance of HOG + Linear SVM [7] object detectors and convolutional neural networks (R-CNN [62], Faster R-CNN [47], YOLO [46] ....) The intersection over the union is simply an evaluation metric. Any algorithm that provides bounding boxes provided as an output can be evaluated using IoU. In Figure 2.7 a visual example of a ground-truth bounding box versus a predicted bounding box:

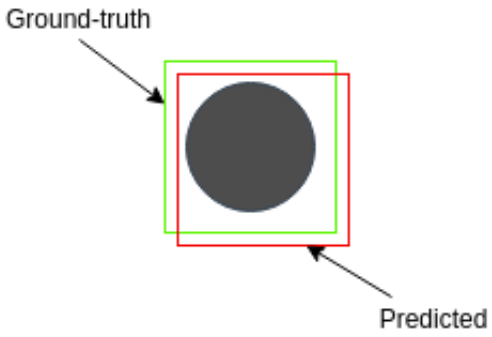


Figure 2.7: Visual example of ground-truth vs predicted bounding box.

Looking at the equation 2.1, Intersection over Union is a ratio. In the numerator, the overlap area between the predicted bounding box and the ground-truth bounding box is calculated. The denominator is the area of union, that is, the area covered by the envisaged boundary box and the ground-truth bounding box. Dividing the overlap area by the union area, we obtain the final score - the Intersection over Union. The example that illustrates the intersection over the union can be seen in Figure 2.8

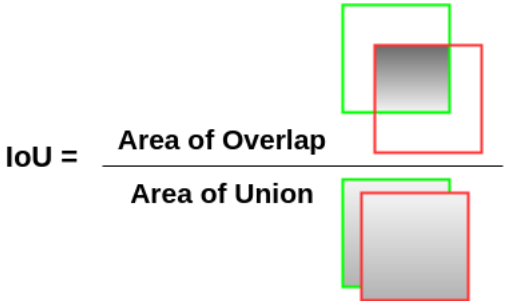


Figure 2.8: Visual example of the Intersection over Union equation.

In Figure 2.9, the quantitative IoU relationships are shown visually. You can see that predicted bounding boxes that strongly overlap with ground-truth bounding boxes have higher scores than those with less overlap. This makes Intersection over Union an excellent metric for evaluating custom object detectors.

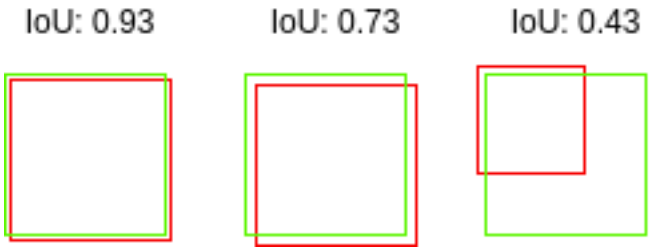


Figure 2.9: An example of computing Intersection over Unions for various bounding boxes.

Both the confidence score and the IoU are used as criteria that determine whether a detection is a true positive or a false positive. The Algorithm 1 shows how:

Algorithm 1: Object detection classification based on the IoU.

input : confidence_score = threshold_a
iou_score = threshold_b
ground-truth
detection

```
1 begin
2     for each detection that has a confidence_score > threshold_a: do
3         among the ground-truths, choose one that belongs to the same class and
          has the highest IoU with the detection
4         if no ground-truth can be chosen or iou_score < threshold_b (e.g., 0.5)
           then detection is a false positive;
5         else detection is a true positive;
```

As shown in the Algorithm 1, we call confidence score, the probability that an anchor box contains an object from a certain class. It is usually predicted by the classifier part of the object detector. The confidence score and IoU score are used as the criteria to determine whether a detection is a true positive or a false positive. Given a minimal threshold on the confidence score for bounding box acceptance, and another threshold on IoU to identify matching boxes, a detection is considered a true positive (tp) if there exists a ground truth such that: confidence score > threshold; the predicted class matches the class of the ground truth; and IoU > $\text{threshold}_{\text{IoU}}$. The violation of any of the last two conditions generates a false positive (fp). In case multiple predictions correspond to the same ground-truth, only the one with the highest confidence score counts as a true positive, while the others are considered false positives. When a ground truth bounding box is left without any matching predicted detection, it counts as a false negative (fn).

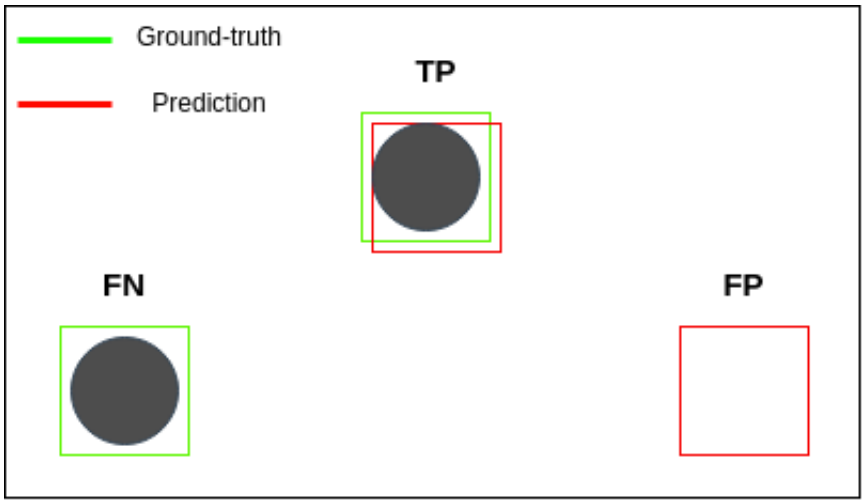


Figure 2.10: Metric object detection for global evaluation of model.

Therefore, from these counters (TP, FP, FN) we can infer the metrics of Precision, Recall and F_1-Score.

- **Precision:** Is defined as the number of true positives divided by the sum of true positives and false positives as the equation 2.2:

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

  A high precision means that most of the predicted boxes had a corresponding ground truth. In other words, the object detector is not producing bad predictions.

- **Recall:** Is defined as the number of true positives divided by the sum of true positives and false negatives as the equation 2.3:

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

high recall means that most of the ground truth boxes had a corresponding prediction. In other words, the object detector finds most objects in the images.

- **F1-Score:** Is the harmonic mean of the precision and recall as the equation 2.4:

$$F_1 = \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.4}$$

F_1-Score is needed when a balance between Precision and Recall is sought. In the case of object detection on production lines, a low precision means that sometimes a part might be absent and the model would not see it, whereas a low recall means that sometimes the part is present and the model raises an alert anyways. For this reason, both a good recall and a precision are required and the choice of using the F_1-Score metric seems appropriate.

2.4.1 Precision x Recall Curve

When predicting the bounding boxes, the models of convolutional neural networks for object detection produce a vector of probabilities corresponding to the confidence over each object class [58]. Therefore, by setting the threshold for confidence scores at different levels, we obtain different pairs of precision and recall. For this reason, the Precision x Recall curve is a good way to assess the performance of an object detector as confidence is changed by plotting a curve for each class of object. In Figure 2.11 you can see an example of this chart.

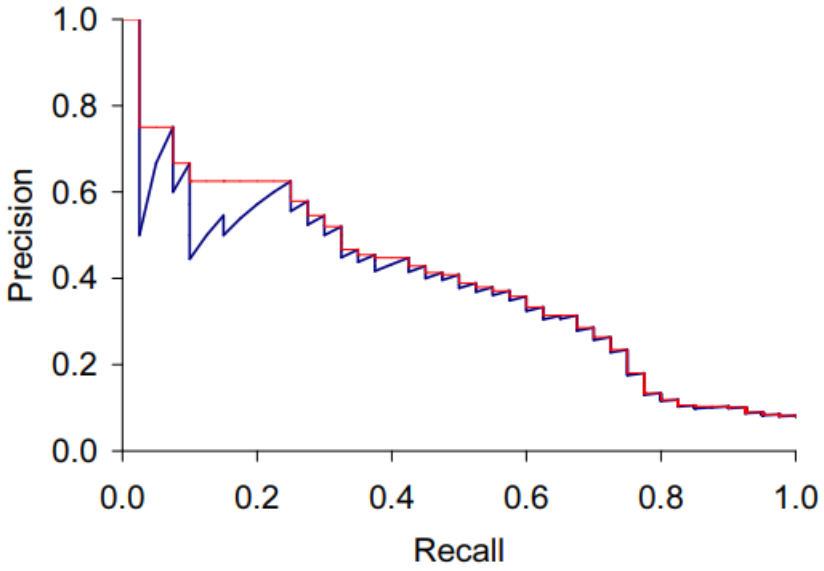


Figure 2.11: An example of computing Intersection over Unions for various bounding boxes.

2.4.2 Average Precision

Another way to compare the performance of object detectors would be to calculate the area under the curve (AUC) of the Precision x Recall curve. However, it becomes difficult to compare different detectors when the AP curves intersect. That is why Average Precision (AP), a numerical metric, is used to compare different detectors. Therefore, AP is the precision average for all recall values between 0 and 1. From there, there are two ways to calculate Average Precision. Both use point interpolation. This metric was well known by [13]

11-points

The first one interpolates 11 points, trying to summarize the shape of the curve in a set of eleven equally spaced recall levels [0, 0.1, 0.2, ..., 1], as described in the equation 2.5.

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, 0.2, \ldots, 1\}} \rho_{interp}(r) \tag{2.5}$$

The precision at each recall level r is interpolated by taking the maximum precision measured for a method for which the corresponding recall exceeds r:

$$\rho_{interp}(r) = \underset{\tilde{r}:\tilde{r}\geq r}{max}\, \rho(\tilde{r}) \tag{2.6}$$

where $\rho(\tilde{r})$ is the measured precision at recall $\tilde{r}$.

All points

Rather than interpolating only 11 points, this method interpolates all n points, as shown in the equation 2.7.

$$\sum_{n=0}(r_{n+1} - r_n)\rho_{interp}(r_{n+1}) \tag{2.7}$$

with

$$\rho_{interp}(r_{n+1}) = \underset{\tilde{r}:\tilde{r}\geq r_{n+1}}{max}\, \rho(\tilde{r}) \tag{2.8}$$

where $\rho(\tilde{r})$ is the measured precision at recall $\tilde{r}$.

AP calculation involves only one class. However, there may be more classes in object detection, so the mean average precision (mAP) is defined as the AP average across all K classes:

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \tag{2.9}$$

The Pascal VOC [19] challenge uses only a single IoU limit of 0.5. However, the COCO [30] challenge defines several mAP metrics using different thresholds, including:

- $\boldsymbol{mAP^{IoU=0.5:0.05:0.95}}$, which is mAP averaged over 10 IoU thresholds (i.e., 0.50, 0.55, 0.60, ..., 0.95) and is the primary challenge metric;
- $\boldsymbol{mAP^{IoU=0.5}}$, which is identical to the Pascal VOC metric
- $\boldsymbol{mAP^{IoU=,.75}}$, which is a strict metric.

Chapter 3

Literature Review

This chapter presents the main works related to object detection using synthetic image generation. The main aspects of the papers are discussed in relation to the implemented models and the methods of generating these synthetic images. Finally, an analysis of the missing characteristics and the points where the various strategies can be improved is presented. So that in the end, the object detector obtains a better generalization performance, cost and time.

The number of articles on object detection based on deep learning is extremely large. However, our focus is based on deep learning object detection using synthetic images. Therefore, each subsection will represent similar works, but using different methods and/or different contexts.

The related works were researched using some bases, among them: Google Scholar [1], IEEE Xplore [2], Springer [3] and ACM Digital Library [4]. The search keywords were: Object Detection CAD Models, Object Detection, Deep Learning Object Detection, Convolutional Neural Networks, Object Detection Synthetic Images, Object Detection Industry 4.0, Object Detection in Assembly Lines.

3.1 Object Detection

According to [17], the ability to detect objects in challenging environments is a key component for many computer vision and robotics tasks. The main current object detectors,

[1] https://scholar.google.com.br/
[2] https://ieeexplore.ieee.org/Xplore/home.jsp
[3] https://www.springer.com/br
[4] https://dl.acm.org/

such as those mentioned in the section 2.3, depend on convolutional neural networks. However, for best performance, they require certain amounts of labeled training data, which are often time consuming and expensive to create. Consequently, the use of synthetic images is very attractive for training object detectors, since labeling is free. Unfortunately, synthetic rendering methods are often unable to reproduce the results produced by their real-world counterparts. This is often referred to as the 'domain gap' between synthetic and real data and the transfer from one to the other usually results in deteriorated performance.

For this reason, the author assesses the freezing of layers responsible for extracting resources in pre-trained generic layers in real images and trains only the remaining layers with OpenGL rendering, allowing training with synthetic images only.

In [57], the author states that estimating from the point of view of the object from 2D images is an essential task in computer vision. However, two problems hinder its progress: the scarcity of training data with point-of-view notes and the lack of powerful resources. For this reason, inspired by the increasing availability of 3D models, the author proposes a framework to solve both problems, combining image synthesis based on rendering and CNNs (Convolutional Neural Networks). For that reason, he believes that 3D models have the potential to generate a large number of high-variation images, which can be well explored by deep CNN with high learning capacity. For this purpose, a scalable and overfit resistant image synthesis pipeline is proposed, together with a new CNN tailored to the point of view estimation task.

With the increasing accessibility of 3D CAD models, an infinite number of training images can be generated for many categories of objects. Because of this, another important work [41], shows that increasing the training data of contemporary models of Deep Convolutional Neural Network (DCNN) with synthetic data can be effective, especially when the actual training data is limited or does not match to the target domain. Most freely available CAD models capture the 3D shape, but they often lack other low-level tips, such as realistic object texture, pose, or background. In a detailed analysis, the author uses synthetic images rendered by CAD to probe the DCNN's ability to learn without these tips. In particular, it is shown that when the DCNN is adjusted in the target detection task, it exhibits a high degree of invariance for missing low level clues, but when pre-trained in the generic ImageNet classification, it learns best when the low level clues are simulated.

So that we can understand the real dimension and importance of this type of work,

researchers from NVIDIA and the University of Toronto at [60], present a training system for deep neural networks to detect objects using synthetic images. To deal with the variability in real-world data, the system depends on the domain randomization technique, in which the parameters of the simulator - such as lighting, pose, object textures, etc. - are randomized in unrealistic ways to force the neural network to learn the essential characteristics of the object of interest. The authors explored the importance of these parameters, showing that it is possible to produce a network with attractive performance using only synthetic data. With additional fine-tuning on real data, the job produces better performance than using only real data. This result opens up the possibility of using cheap synthetic data to train neural networks, avoiding the need to collect large amounts of hand-labelled real-world data or to generate high-fidelity synthetic worlds - both of which remain bottlenecks for many applications.

According to [48], an important insight from this type of approach is that the images generated synthetically must be similar to the real images, not in terms of image quality, but in terms of resources used during detector training. Therefore, in this same work, it is shown that in the context of detecting drones, airplanes and cars, the use of such synthetically generated images produces significantly better performances than simply disturbing real images or even synthesizing images in such a way that they appear very realistic, as it is usually done when only limited amounts of training data are available.

A training set can also be simulated by applying small deformations and adding noise to the images it contains. Therefore, there are several techniques for generating synthetic data. Synthetic data were used to study the effects (for example, object texture, color, 3D pose and background) on the performance of the image classifier, according to [1]. There is even a hypothesis that advances in computer graphics may positively affect the quality of synthetic data, so that the photographs generated synthetically will be almost indistinguishable from those in the real world [42].

An important tool in the generation of virtual images is Blender. As a result, in [21], the project used Blender computer graphics modeling software with customizable settings that allow users to design detailed three-dimensional scenes and objects. A plug-in was developed to automate the generation of variations in the scene during rendering and combine them with various environmental backgrounds. Because the software has full access to the scene data, the automated labeling of the generated images can produce a metadata file containing information such as the object's bounding box, the object's position in the scene and the position where the image was occupied. Although this

approach allows the user to be removed from the manual data collection and labeling processes, it still requires the user to initially create the scene.

3.2 Industrial Approaches

All the works mentioned above use generic objects such as cars, buses, airplanes, toys, boats, among others. However, due to the theme of this work, it is important that we mention the works that have a focus on industrial applications.

Showing the real importance of public datasets as a vital tool for computer vision, *MVTec Industrial 3D Object Detection Dataset* [11] develops a public dataset for 3D object detection and pose estimation with a strong focus on objects , configurations and requirements that are realistic for industrial configurations. Unlike other 3D object detection data sets that often represent scenarios of everyday life or mobile robotic environments, the work cited models industrial waste collection and object inspection tasks that often face different challenges. In addition, the evaluation criteria are focused on practical aspects, such as execution times, memory consumption, useful measures of correction and precision. The data set contains 28 objects with different characteristics, organized in more than 800 scenes and labeled with about 3500 rigid 3D transformations of the object's instances as fundamental truth. Two industrial 3D sensors and three high-resolution grayscale cameras look at the scene from different angles, allowing you to evaluate methods that operate on a variety of different modalities. The author initially evaluates 5 different methods in the data set. Although some show good results, there is a lot of room for improvement.

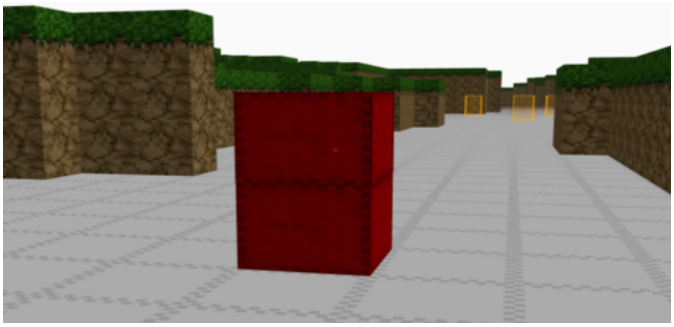
More recently, in [69], a method of detecting defects of tiny parts in real time was developed. The author considers the important influences of the properties of the tiny parts and the environmental parameters of a defect detection system in its stability. Second, it establishes a correlation model between the detection system coefficient of the parts system and the speed of movement of the conveyor. Finally, the author proposes a defect detection algorithm for tiny parts, based on a single network of short detectors (SSD) and deep learning. Finally, it combines an industrial real-time detection platform with the lost detection algorithm for mechanical parts based on intermediate variables to solve the lost detection problem. A 0.8 cm darning needle was used as an experimental object.

3.3 CAD Models

First commercial CAD programs came up in the 1970s and provided functions for 2D-drawings and data archival [18] and thereby evolved into the main engineering design tool [31]. As a result, graphical or synthetic CAD models have also been used to support multiview detection, as exemplified in [70]. In Figure 3.1, we can see some examples of using CAD models in different scenarios.

(a) Example of CAD model in object detection scenario [41]

(b) Example of CAD model in industry scenario [5]



(c) Example of CAD model in games scenario [27]

Figure 3.1: CAD models examples for different scenarios approaches

In this context, CAD models have been improved, integrating local and global computer vision data to represent an object not only geometrically, but also in terms of computer vision. These models can provide a scalable solution for intelligent and automatic object recognition, tracking and augmentation based on generic object models [3].

Therefore, it is possible to observe that the use of synthetic data has a long history in computer vision. Among the first attempts, 3D models were used as the primary source of information to build object models [41]. More recently, some have used 3D CAD models as their only source of labeled data, but have limited their work to a few categories, such as cars and motorcycles [30, 13].

Several works, such as those previously mentioned, use training methods for object detection and recognition. Some of them even use synthetic images for this. Therefore, this subsection is intended for a brief discussion of the points covered by the related works, their results and approaches.

After all the discussion of the results and methods used by the related works, we can see that there are different approaches for object detection, such as using synthetic images or not, how many images were necessary to obtain a certain result, context of the generation scenes, synthetic images, types of objects to be detected, etc. These works, certainly, served as a basis for developing our systematic approach. However, our proposal is different in some ways, which will be presented in the next chapter.

Therefore, there are different detection tasks, with difficulties that vary between them. However, we can mention computer vision problems, such as objects from different points of view, illuminations, interclass variations. In addition, object rotation, scale changes, precise location, speed of detection, etc.

Chapter 4

Systematic Approach

In this chapter, the systematic approach proposed for object detection developed in this master's thesis is discussed. In addition, during the development of the systematic approach, a dataset [1] was elaborated, exactly for the purposes of this work, with the objective of public disclosure to other users and researchers to use the available data. Considering the diversity of researched works, until now, this was the only one that proposed the development of a specific systematic approach for object detection using synthetic images applied to the automation of industrial processes.

4.1 Overview

The systematic approach consists of three parts: synthetic image data set generation with automatic bounding box labelling, network training with few images and the complete industrial process. First, a custom code (using free and open-source 3D computer graphics software tools) is used to generate labeled training images containing the CAD model rendered in context. Then, a pre-trained object detection model is fitted to the generated data set. Subsequently, the model can be used for inference in real images. Finally, all the previous steps are brought together in a single, fully systematic industrial process. The quality of the training pipeline is assessed by computing standard object detection metrics in the actual images, and these results are used to test and evaluate different values of the user-defined parameters of the training process. In subsequent sections, we explain our approach in more detail.

[1] `https://github.com/igorgbs/systematic_approach_cad_models`

4.2 Synthetic Image Generation

In order to generate a synthetic image sample, our process requires several elements. First, a CAD model of the object of interest as well as several other industrial CAD models need to be available. After that, the synthetic image generation is divided into some substeps. A floor and a table are created and some distractors are sampled. Using physics simulation, the distractors are dropped from a random height on the table. The position of the object of interest is also randomly sampled. Once the 3D scene is created, textures and colors are sampled for the backgrounds and the distractors and the entire scene is textured. Light sources and cameras are also sampled and placed randomly. Constraints on the camera pose are applied, in order to ensure that the object appears in the camera view. Once the scene has been created, the rendering occurs and generates an image. By removing the light sources and making the object of interest a light source itself, we can generate another image which can be used for bounding box labelling. This procedure is necessary because even if we know the location of the object, it can be partly hidden by distractors and thus distort the labelling. The process for creating these images can be seen in Figure 4.1.

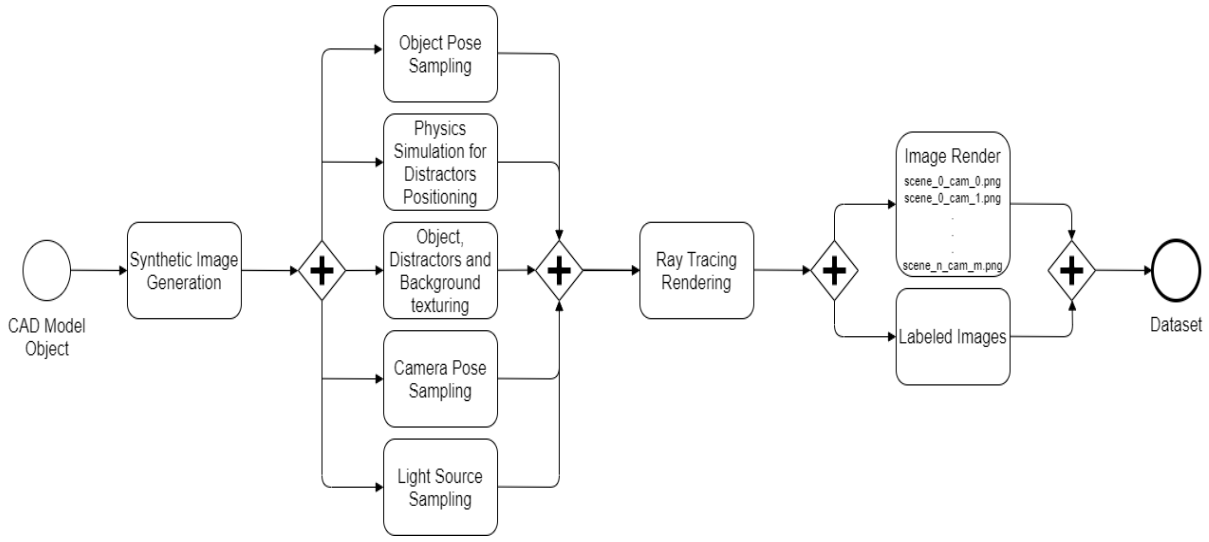


Figure 4.1: Synthetic image generation process

An example of the synthetic image dataset generation process can be seen in Figure 4.2.

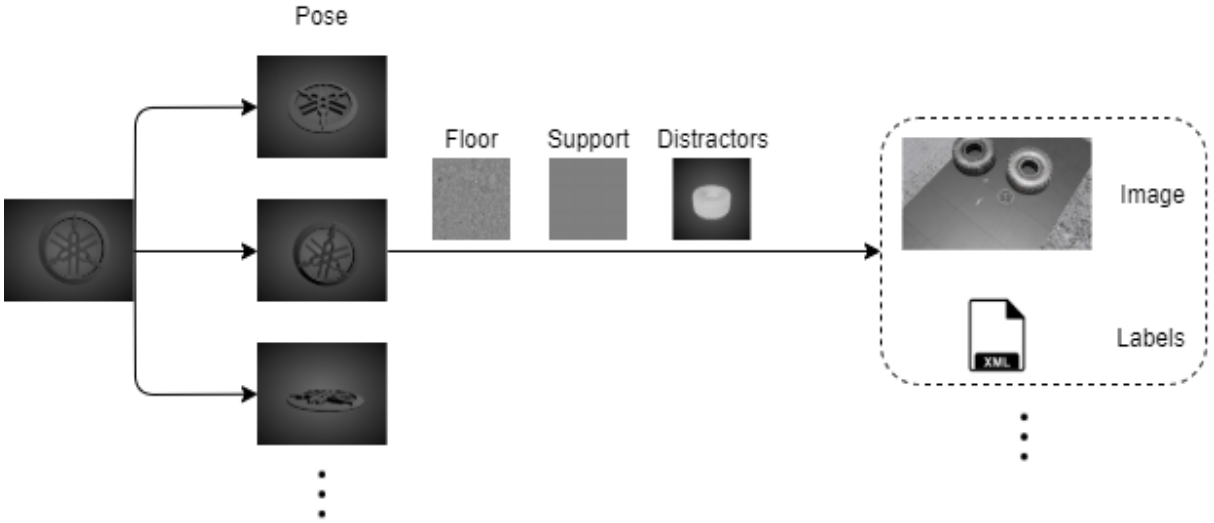


Figure 4.2: Synthetic image generation process example

4.2.1 Implementation

For the automatic generation of the training images (with pose sampling, rotation, physics simulation, light sampling...), the software Blender [6] was used. Blender is a powerful software for 3D creation which includes features such as modeling, rigging, simulation and rendering. We chose to use Blender because it presents the advantage of having a good Python API, it is open-source and it has good GPU support.

In the experiments of this thesis, several objects of interest were used in order to validate our systematic approach. For each object generated synthetically, there are corresponding real objects. The other objects that make up the scene serve as distractors to help the model focus on the right object. The CAD models for the distractors and main objects are gathered from the Grabcad website [2]. Then different textures for the different distractors as well as for the background need to be gathered. In practice, we collected these textures from the Poliigon website[3]. Finally, we need to manually reproduce the color of the object of interest. The code for the synthetic images generation was originally developed by Joris Guérin (co-advisor of this work) and adapted for this work and can be seen at github [4].

[2] `https://grabcad.com/`
[3] `https://www.poliigon.com/`
[4] `https://github.com/igorgbs/systematic_approach_cad_models`

4.3 Hyperparameter Tuning

The Blender script for generating images has many hyperparameters that must be chosen before using it, therefore, if necessary, a procedure for selecting parameters to optimize the adjustable hyperparameters of the systematic approach. The adjustable parameters used are the resolution of the generated training images, the number of camera poses per generated scene, the number of generated scenes, the maximum number of objects studied per scene, the maximum number of distractors to be included in each scene and the number of textures.

Hyperparameter tuning is a crucial step in machine learning practice. Recently, it was shown that the state of the art on image classification benchmarks can be improved by configuring existing techniques better rather than inventing new learning paradigms and hyperparameter tuning is often carried out by hand, progressively refining a grid over the hyperparameter space [2]. Therefore, after each combination in the generation of images, there comes the training phase of the model that uses convolutional neural network in the synthetic images that were generated. The model also has hyperparameters to be adjusted, such as sampling of the training and test subsets, pre-trained object detection model, number of steps, batch size, momentum optimizer value and learning rate. These combinations are used to identify the best hyperparameters for the systematic approach.

4.4 Training

Once a synthetic data set of images labelled with bounding boxes coordinates has been generated, it can be used to train an object detector. Here, we describe the main steps for the training process: preparing workspace, annotating images and configuring training model, using the TensorFlow OD API.

4.4.1 Preparing Workspace

The first part consists in organizing the training directories in order to structure the folders containing the training image files, the label files containing the bounding boxes coordinates and the reference file containing the names of the classes to detect. This step is carried out automatically by our script.

4.4.2 Annotating Images

Two types of annotation files are needed. First, for each training image, there must be an associated label file containing the coordinates of all bounding boxes encompassing the objects of interest. Every bounding box is described by the object class name and the following coordinates: (x_{min}, x_{max}, y_{min}, y_{max}). These coordinates are referenced according to the image resolution. As explained earlier, the label files are generated automatically by the Blender script for the training images.

Besides the training set, bounding box annotation files are also needed for the test images. In our case, we want to validate that an object detector trained on synthetic images can generalize to real world industrial cases. For this reason, the set of test images contains real images, that is, photos of the analyzed objects. The files structure follows the same organization as the training set. However, for real images, the ground truth bounding boxes cannot be generated with Blender and must be made manually. To do this, we use a software called LabelImg [5]. This application allows us to draw and save the annotations of each image as xml files in the PASCAL VOC format [19], which is also the format used by ImageNet [9]. After generating all the annotations in xml format, it is necessary to process these files in the csv format, then later, convert them into .record[6].

4.4.3 Configuring Training Model

In this work, we used one of the pre-trained models provided by the TensorFlow Object Detection API. By using this transfer learning approach, we start with a model that already knows basic feature extraction skills and is less likely to overfit the synthetic data sets. Indeed, the diversity that we can create with Blender is limited as we cannot get an infinite amount of textures and distractors, and the diversity already encountered by the network during pre-training can help reduce overfitting. In addition, using a network pre-trained on real images can prevent the network from learning detection features that depend too much on the generation procedure.

There exists several models in the TensorFlow object detection model zoo. More information on the performance of the detection, as well as the reference execution times, for each of the available pre-trained models, can be found on the Github page of the API[7].

[5] https://github.com/tzutalin/labelImg

[6] https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html\#exporting-a-trained-inference-graph

[7] https://github.com/tensorflow/models

In practice, the model used in this paper is the *faster_rcnn_inception_v2_coco* model, which provides a good trade-off between performance and speed. This model was selected following the results of experiments on another data set, which was used for practical implementation of an industrial application.

Faster R-CNN, the model used in this work, takes as input an entire image and a set of object proposals. The network first processes the whole image with several convolutional and max pooling layers to produce a convolutional feature map. Then, for each object proposal, a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map.

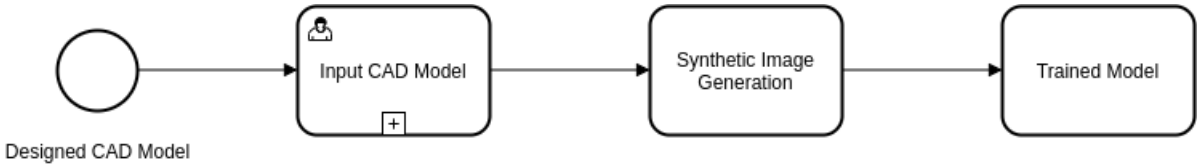
Each feature vector is fed into a sequence of fully connected layers that finally branch into two sibling output layers: one that produces softmax probability estimates over K object classes plus a catch-all "background" class and another layer that outputs four real-valued numbers for each of the K object classes. Each set of 4 values encodes refined bounding-box positions for one of the K classes [15].

4.5 Industrial Process Scenario

Quality assessment in many production processes typically relies on human inspections due to a lack of reference data and an effective method to classify defects in a systematic way. Recently, the real-time, automated approach for product quality assessment has been regarded as an important aspect for smart manufacturing applications, such as in the automotive industry [38].

With that, the real-time object detection and tracking have shown to be the basis of intelligent production for industry 4.0 applications. It is a challenging task because of various distorted data in complex industrial setting. In this setting, a simple and highly efficient method is the key for trade off low-computation and robustness during separating various objects from cluttered background [34]. For this reason, this section describes the part of the work aimed at applying the systematic approach in the industrial environment, through a process.

Now, imagine a real production line scenario, where a certain company needs to check some quality process, detecting objects about a certain part, product or even an entire part of the production process. Based on this scenario, our systematic approach involves a process diagram, where it tries to fill this automation gap in quality processes. The diagram can be seen in Figure 4.3.



(a) First workflow of automation process

(b) Second workflow of automation process

Figure 4.3: Automation of the quality assessment process in the industry through object detection

As in any industry, the first step in product development is in product design. Thus, the parametric CAD software is the primary development tool for the design engineer during the product development process [31]. Computer-aided design (CAD) plays a central role in automotive development. 3D-CAD models not only provide geometry information, but also serve as a basis for configuration of modules and systems, as well as for different simulation and verification processes. Product assemblies within CAD-platforms include structure-related information, are used for digital mock-up (DMU) investigations and provide lots of data for production and manufacturing engineering [18]. Taking advantage of this process (product design) inherent to all types of manufacturing processes, all model training is based on this object already designed by the engineering team, applied to a specific context. Another advantage of using the CAD model for training the object detector is that the implementation of the quality control process can be done even before the production line is operational, using only digital elements.

After the CAD model is ready, the product designer (in this case - the engineer) is enough to insert it into the developed system. Once the CAD model has been inserted, synthetic scenes with the object in question will be generated and soon after that, the system itself will take care of training the model for that object applied to the industrial context in question, thus completing the first workflow of automation process. After that, the system's output is a trained model capable of detecting the presence of the object in the production line, represented by the second workflow of automation process.

Chapter 5

Experimental Evaluation

The experiments that were carried out using the neural network model used in this work for the evaluated objects, are described in this chapter. The parameters and hyperparameters adopted for the object detection scenarios are discussed after the various tests performed. The different results obtained are compared and graphs are drawn up to better visualize the final result for each object inserted in the production lines domain.

5.1 Image Generation

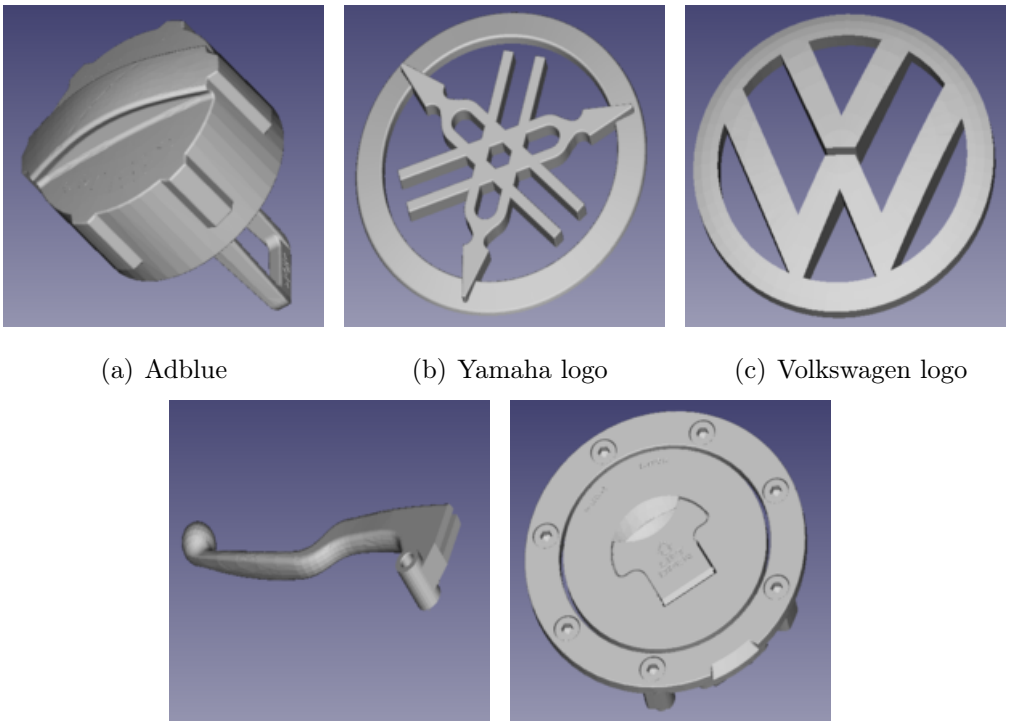
In this section, we evaluate different adjustable parameters to generate synthetic images based on CAD models. In addition, we evaluated different parameters to train the deep object detection model. Therefore, we carry out a parameter selection procedure, as described in the section 4.3. The chosen values for the experiments are as follows in Table 5.1:

In addition, five objects were chosen for the experiment. The CAD models of these

Parameters	**Values**
Resolution Train	960x540, 640x480, 1280x720
Camera poses	2, 5, 20
Number of scenes	20, 50, 200
Number of (main) objects	0 or 1, 1, 1 to 4
Number of distractors	0, 5, 20
Floor Textures	1, 3, 7
Distractor Textures	1, 3, 7
Support Textures	1, 3, 7

Table 5.1: Parameters set for Blender image generation.

objects can be seen in Figure 5.1.



(a) Adblue (b) Yamaha logo (c) Volkswagen logo

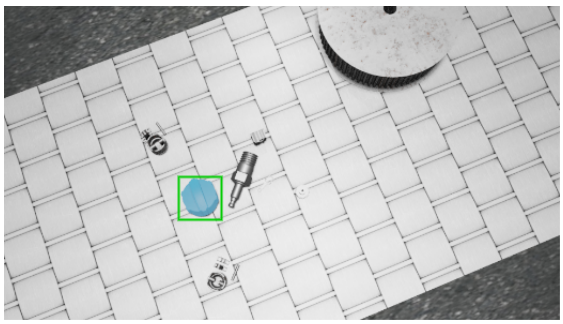
(d) Clutch lever (e) Fuel cap

Figure 5.1: CAD models of the objects chosen for the experiment.

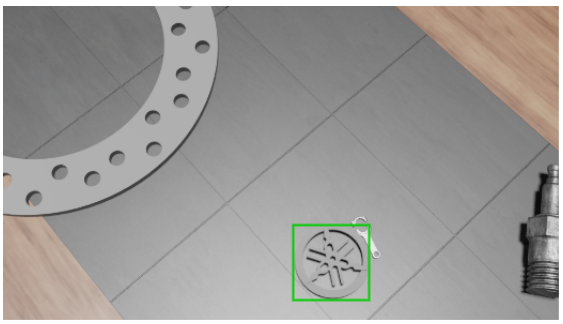
From this set of parameters, we chose an object to perform combinations in order to achieve the best scenario for generation as shown in Table 5.2. These scenarios were evaluated in a test set of real images. Thirty parameter combinations were made, with 10 samples per combination, since the neural network started with random weights. With these preliminary experiments on a dataset of a specific object (Adblue), we concluded that the number of textures used for the floor, the distractors and the support must be adjusted to the maximum number of textures available (in our case 7 for the floor and 6 for the other two), creating a more complete and complex scenario for object detection. From the chosen objects, the images are generated, as shown in Figure 5.2.

Parameters	Values
Resolution Train	960x540
Camera poses	5
Number of scenes	20
Number of (main) objects	1
Number of distractors	20
Floor Textures	7
Distractor Textures	7
Support Textures	7

Table 5.2: Final parameters chosen for imaging

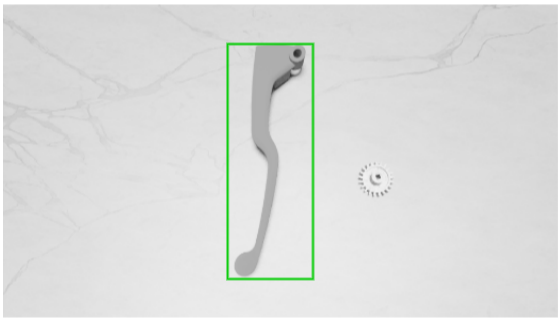


(a) Adblue scene example

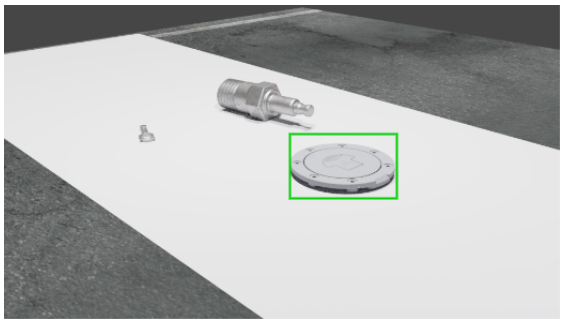


(b) Yamaha logo scene example

(c) Volkswagen logo scene example



(d) Clutch lever scene example



(e) Fuel cap scene example

Figure 5.2: Example of image generated using our custom Blender script.

Parameters	**Best Case**	**Worst Case**
Resolution	960x540	960x540
cam_poses	5	5
n_scenes	20	20
n_images	100	100
n_distractors	20	0
Generation Time	1257.30 sec	749.92 sec
n_samples	10	10
Precision %: Avg *(Std.Dev)*	83.78 *(7.98)*	30.09 *(6.11)*
Recall %: Avg *(Std.Dev)*	88.85 *(4.88)*	92.00 *(3.53)*
F1-Score %: Avg *(Std.Dev)*	85.92 *(3.74)*	45.02 *(7.00)*

Table 5.3: Best and worst hyperparameters configurations obtained and their corresponding results.

In Table 5.3, we can see that distractors are an essential element in our proposed image generation pipeline (e.g., Adblue object). The adblue is the only one that has a different coloring, because it is in this color tone that it appears in real images. The other objects are presented in this shade of gray, as shown in Figure 5.2. In fact, by removing them, we can see a drop of about 47.6 % in the F1 score, on average among the 10 experiment samples. We also tried combinations with few distractors, but the F1 score results dropped significantly. This makes sense when we think that the actual images evaluated also contained several distractors in their scenes. Besides that, in the worst case, Recall has a high value because there are few false negatives (FN) in its prediction. This is due to the fact that the model is always identifying an object in the scene, causing there to be few scenes where the object is not identified (case for FN). However, these predictions have a low level of precision.

During these tests of image generation parameter adjustments, we noticed that very high image resolutions, such as Full HD (1920x1080) and 4K (3840x2160) could not be generated, due to the GPU's memory overflow effect. These experiments were conducted on a Nvidia Quadro P5000 GPU and a 2.90GHz Intel Xeon E3-154M v5 processor (16 GB of RAM).

5.2 Datasets

Now that the optimal hyperparameters were identified with the previous experiments, we want to compare the synthetic image generation approach with different kind of training approaches. The general objective was to cover the three test cases: synthetic images

datasets, real images datasets and hybrid datasets. For this, in addition to the generation of synthetic images datasets, photos were taken from videos made manually and also from videos taken from the internet. The idea was to cover scenarios closer to the reality found in the industry. Each dataset was built for a given object, therefore adding up to a total of 5 sets (one for each object).

5.3 Models Tests

In this section, we evaluate the results of all tests performed in terms of the following metrics: precision, recall, F1-score and AP. In addition, we show the loss functions during the training of each object. The chosen stopping point was 10,000 steps. This decision was made, because in the time vs. precision relationship, the values achieved for the metrics used already showed a significant result, i.e., a good precision, recall and f1-score with a low amount of steps. In this way, each test was carried out with 10 training samples, so that the initial randomness factor of the weights could not be taken into account. Therefore, as mentioned earlier, three test cases were considered: synthetic images, hybrid images and real images.

In Table 5.4 we can see the list of tests performed. All tests were performed with 960x540 resolution images and evaluated on a dataset of 380 previously labeled images for each object, respecting, obviously, the separation between training and test dataset. In addition, the confidence_score and iou_score parameters were set to 0.9 and 0.5, respectively.

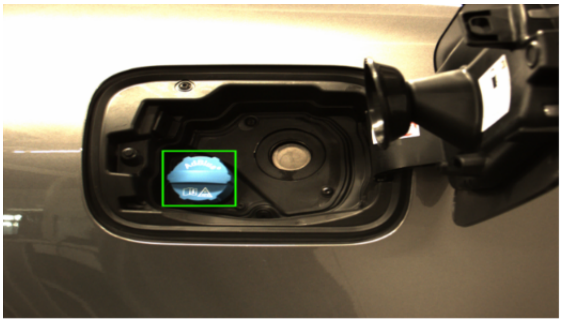
Scenario	**Object**	**Number of Synthetic Images**	**Number of Real Images**
Synthetic	Adblue	100	0
	Yamaha logo	100	0
	Volkswagen logo	100	0
	Fuel cap	100	0
	Clutch lever	100	0
Hybrid	Adblue	50	50
	Yamaha logo	50	50
	Volkswagen logo	50	50
	Fuel cap	50	50
	Clutch lever	50	50
Real	Adblue	0	100
	Yamaha logo	0	100
	Volkswagen logo	0	100
	Fuel cap	0	100
	Clutch lever	0	100

Table 5.4: Training dataset for all scenery tests

Otherwise, in Table 5.5, we can see the number of real images test datasets for each object. In addition, an example real image of each object can be seen in Figure 5.3. The ground-truth of the real images was done manually in order to be able to evaluate the final performance of the model.

Object	Images
Adblue	380
Yamaha logo	380
Volkswagen logo	380
Fuel cap	380
Clutch lever	380

Table 5.5: Real images test dataset



(a) Adblue scene example

(b) Yamaha logo scene example

(c) Volkswagen logo scene example

(d) Clutch lever scene example

(e) Fuel cap scene example

Figure 5.3: Example of real images for each object.

5.3.1 Synthetic Images Evaluation

With 100 images generated through Blender for each object and automatically labeled, 10 test samples were performed to obtain the results described in this section, not including the intermediate experiments, with other parameters and reduced number of samples.

The average and standard deviation of the metrics (precision, recall and F1-score) for each object can be seen in the Tables 5.6 and 5.7. The confusion matrices for the best results for each object can be seen in the Table 5.8. The corresponding Average Precision values [19], were also calculated and are reported in Precision x Recall curves for the IoU ≥ 0.5. The curves for the best results for each object can be seen in Figures 5.4, 5.5, 5.6, 5.7 and 5.8. The curves were plotted with python, using the algorithm developed in [39]. In Figures 5.9, 5.10, 5.11, 5.12 and 5.13, we can observe the loss functions for the best results for each object. The graphs of the loss function were plotted using the TensorBoard - TensorFlow API. The y axis represents the loss and the x axis represents the number of steps. The goal is to minimize the loss function. In other words, it means that the model is making fewer mistakes.

Object	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Adblue	85.11	80.00	81.93
Yamaha logo	78.06	96.22	85.19
Volkswagen logo	0	0	0
Fuel cap	0	0	0
Clutch lever	0	0	0

Table 5.6: Average of test samples for each object with synthetic images

Object	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Adblue	10.49	4.46	3.88
Yamaha logo	16.24	4.29	11.15
Volkswagen logo	0	0	0
Fuel cap	0	0	0
Clutch lever	0	0	0

Table 5.7: Standard deviation of test samples for each object with synthetic images

It is important to note that the values filled with zero in Tables 5.6, 5.7 and 5.8 occurred due to overfitting for these corresponding objects in the described test scenario. Thus showing that in some cases the fully synthetic approach is not working. This justifies the use of the hybrid approach in the later sections.

Object	contains	not contains
	adblue	not adblue
adblue	219	24
not adblue	16	-
	yamaha logo	not yamaha logo
yamaha logo	264	11
not yamaha logo	6	-
	volkswagen logo	not volkswagen logo
volkswagen logo	0	0
not volkswagen logo	0	-
	fuel cap	not fuel cap
fuel cap	0	0
not fuel cap	0	-
	clutch lever	not clutch lever
clutch lever	0	0
not clutch lever	0	-

Table 5.8: Confusion matrix for better results of each object with synthetic images

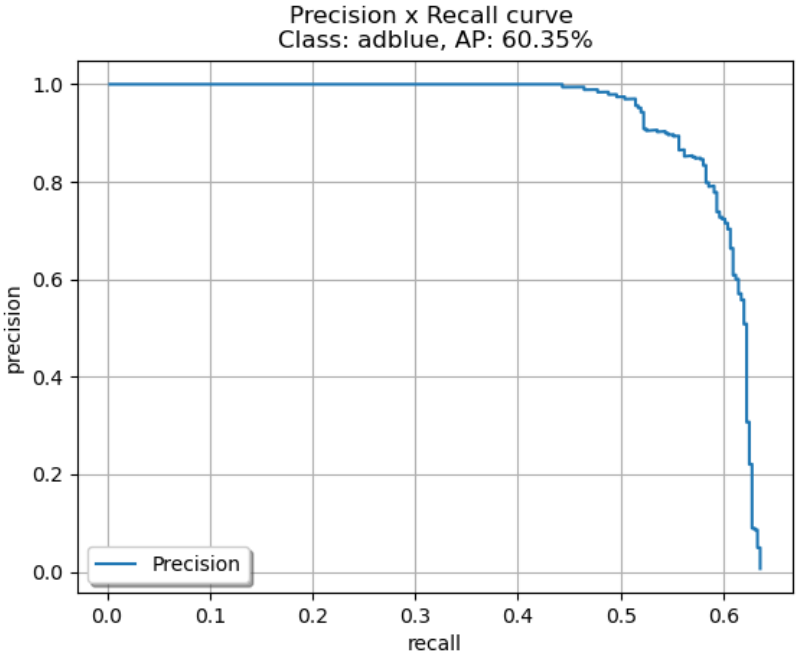


Figure 5.4: Precision x Recall curves for the better test of adblue object with synthetic images

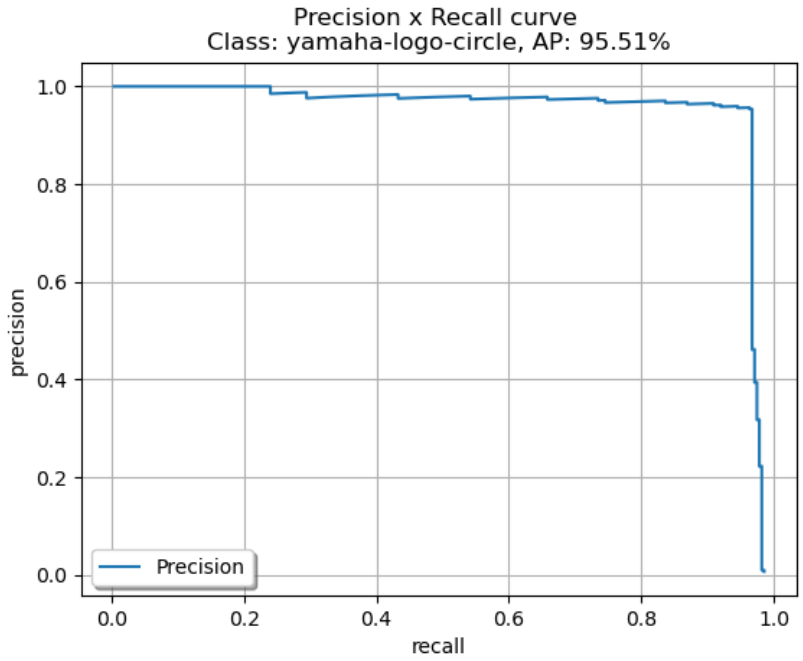


Figure 5.5: Precision x Recall curves for the better test of yamaha_logo object with synthetic images

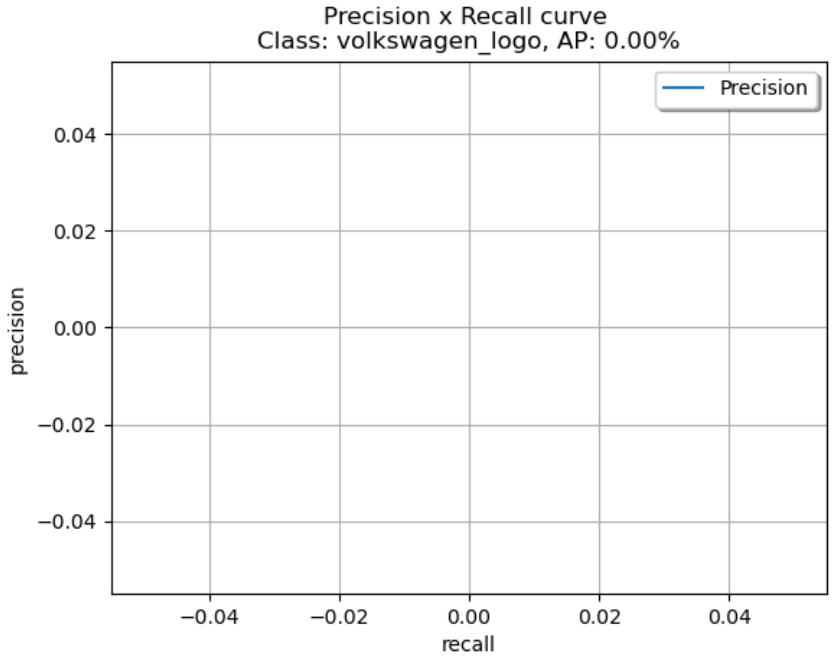


Figure 5.6: Precision x Recall curves for the better test of volkswagen_logo object with synthetic images

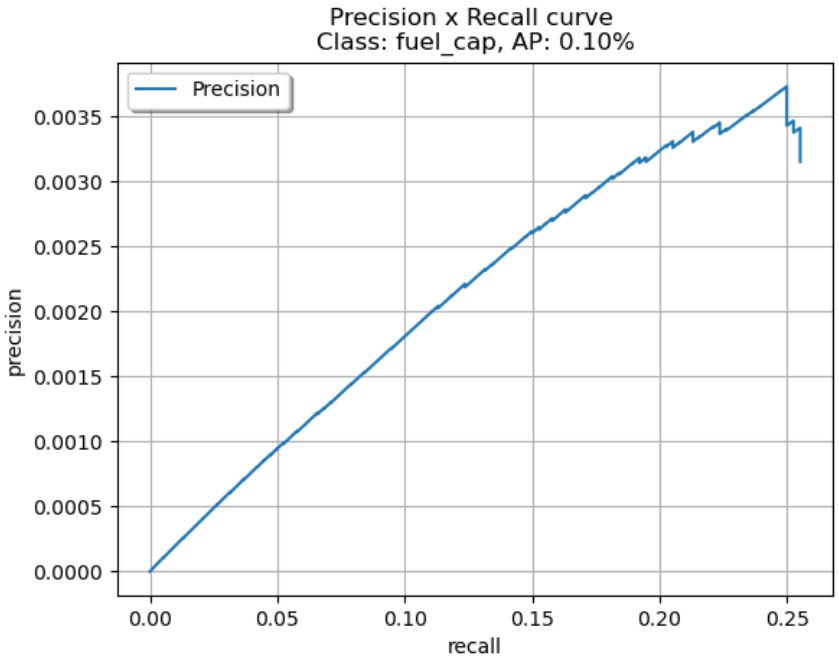


Figure 5.7: Precision x Recall curves for the better test of fuel cap object with synthetic images

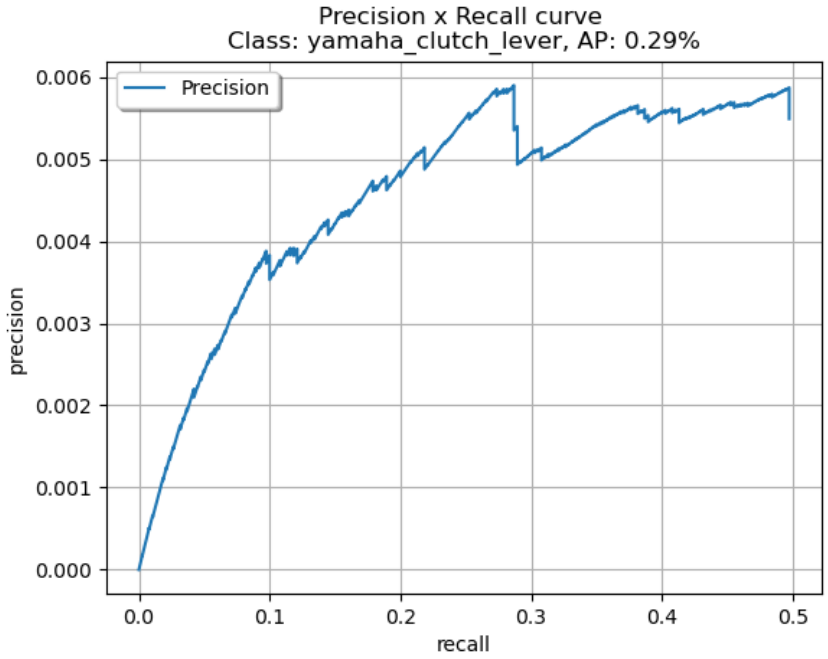


Figure 5.8: Precision x Recall curves for the better test of clutch lever object with synthetic images

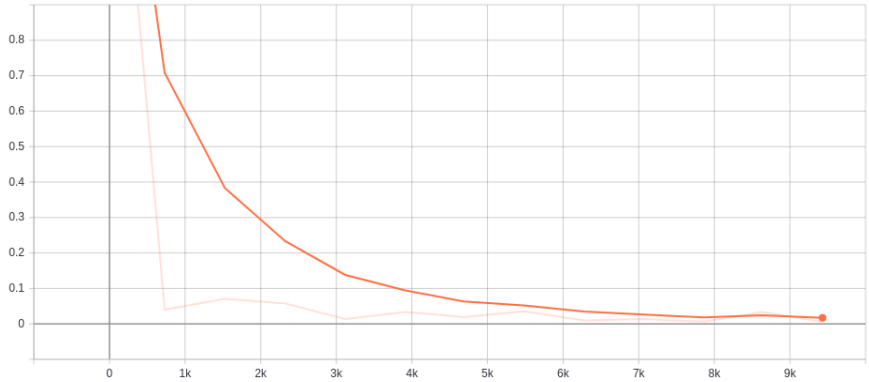


Figure 5.9: Loss function for better result of adblue object with synthetic images

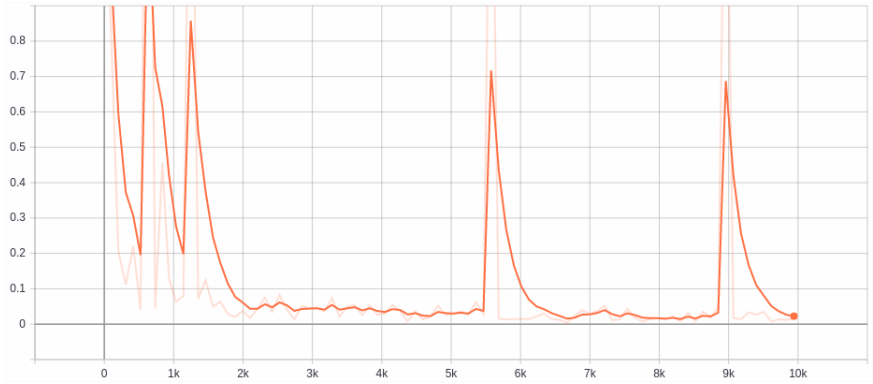


Figure 5.10: Loss function for better result of yamaha logo object with synthetic images

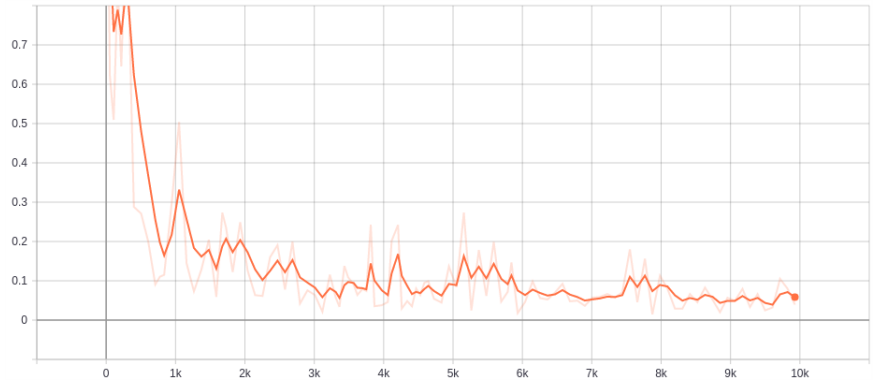


Figure 5.11: Loss function for better result of volkswagen logo object with synthetic images

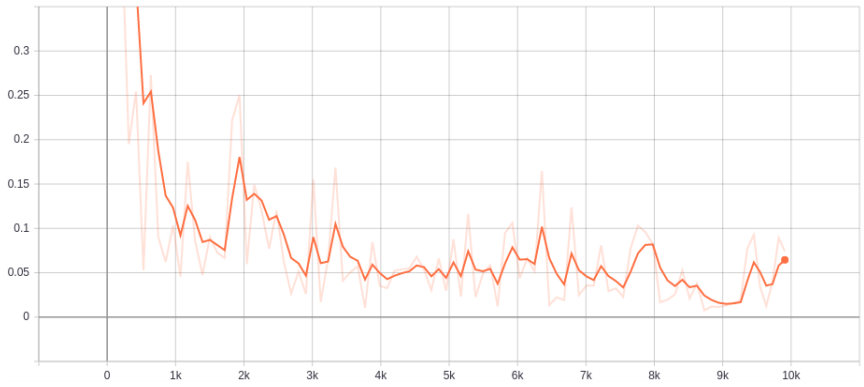


Figure 5.12: Loss function for better result of fuelcap object with synthetic images

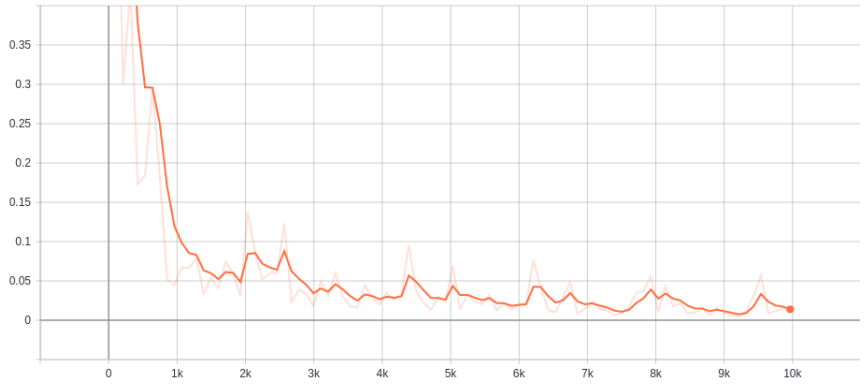


Figure 5.13: Loss function for better result of clutch lever object with synthetic images

5.3.2 Hybrid Images Tests

With 50 images generated by Blender for each object automatically labeled and 50 real images manually labeled, 10 test samples were performed to obtain the results described in this section, not including the intermediate experiments, with other parameters and reduced number of samples.

The average and standard deviation of the metrics (precision, recall and F1-score) for each object can be seen in the Tables 5.9 and 5.10. The confusion matrices for the best results for each object can be seen in the Table 5.11. The corresponding Average Precision values [19], were also calculated and are reported in Precision x Recall curves for the IoU ≥ 0.5. The curves for the best results for each object can be seen in Figures 5.14, 5.15, 5.16, 5.17 and 5.18. The curves were plotted with python, using the algorithm developed in [39]. In Figures 5.19, 5.20, 5.21, 5.22 and 5.23, we can observe the loss functions for the best results for each object. The graphs of the loss function were plotted using the TensorBoard - TensorFlow API. The y axis represents the loss and the x axis represents

the number of steps. The goal is to minimize the loss function. In other words, it means that the model is making fewer mistakes.

Object	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Adblue	87.13	44.64	58.70
Yamaha logo	95.04	95.79	95.31
Volkswagen logo	98.27	62.98	75.06
Fuel cap	94.22	91.74	92.84
Clutch lever	73.38	29.11	41.17

Table 5.9: Average of test samples for each object with hybrid images

Object	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Adblue	5.17	7.26	6.30
Yamaha logo	3.79	4.33	2.65
Volkswagen logo	3.82	18.61	13.36
Fuel cap	2.86	5.94	3.38
Clutch lever	21.73	4.03	7.37

Table 5.10: Standard deviation of test samples for each object with hybrid images

Object	**contains**	**not contains**
	adblue	not adblue
adblue	139	33
not adblue	96	-
	yamaha logo	not yamaha logo
yamaha logo	261	36
not yamaha logo	9	
	volkswagen logo	not volkswagen logo
volkswagen logo	171	0
not volkswagen logo	30	-
	fuel cap	not fuel cap
fuel cap	358	12
not fuel cap	22	-
	clutch lever	not clutch lever
clutch lever	120	15
not clutch lever	200	-

Table 5.11: Confusion matrix for better results of each object with hybrid images

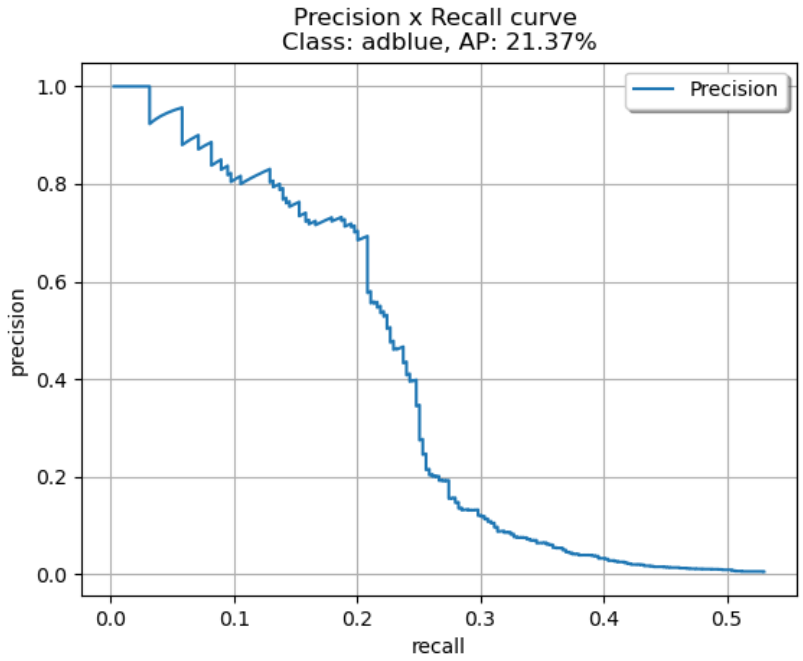


Figure 5.14: Precision x Recall curves for the better test of adblue object with hybrid images

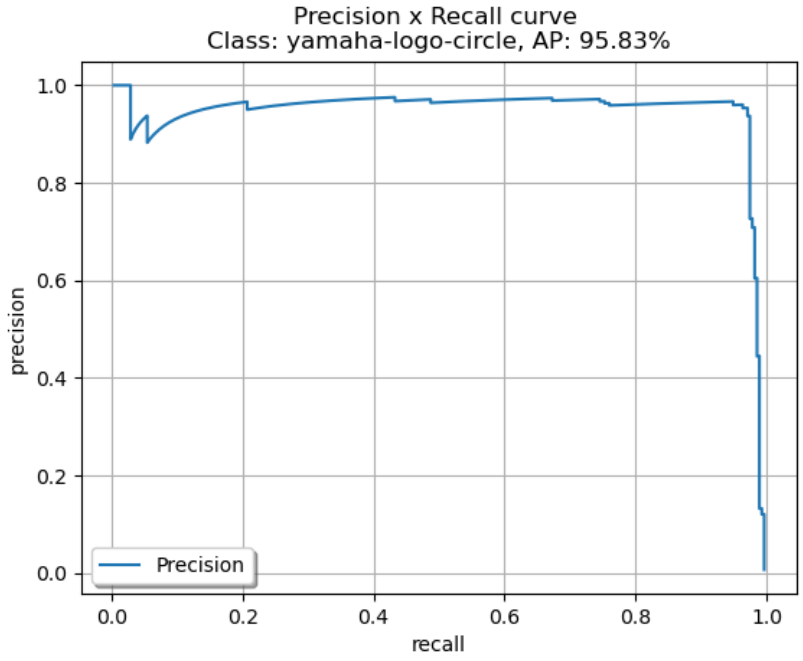


Figure 5.15: Precision x Recall curves for the better test of yamaha_logo object with hybrid images

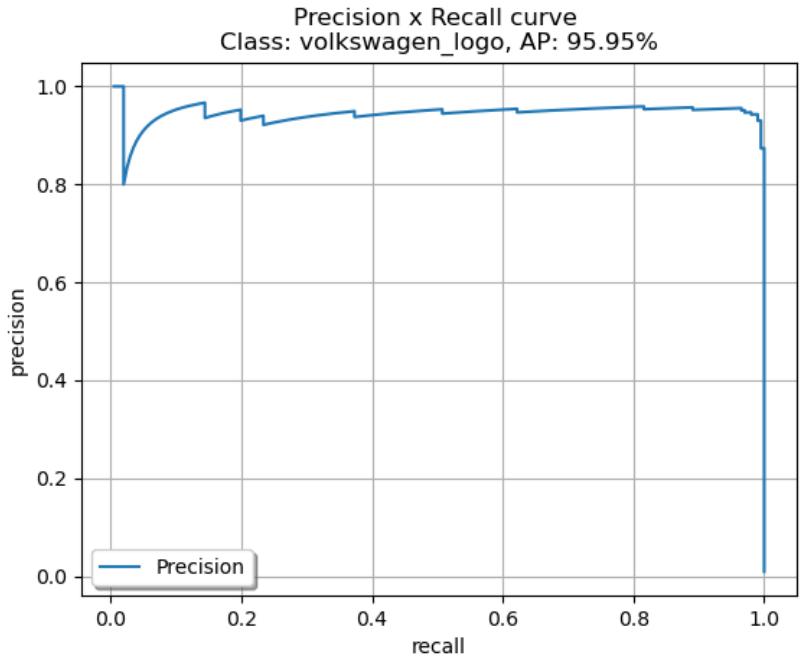


Figure 5.16: Precision x Recall curves for the better test of volkswagen_logo object with hybrid images

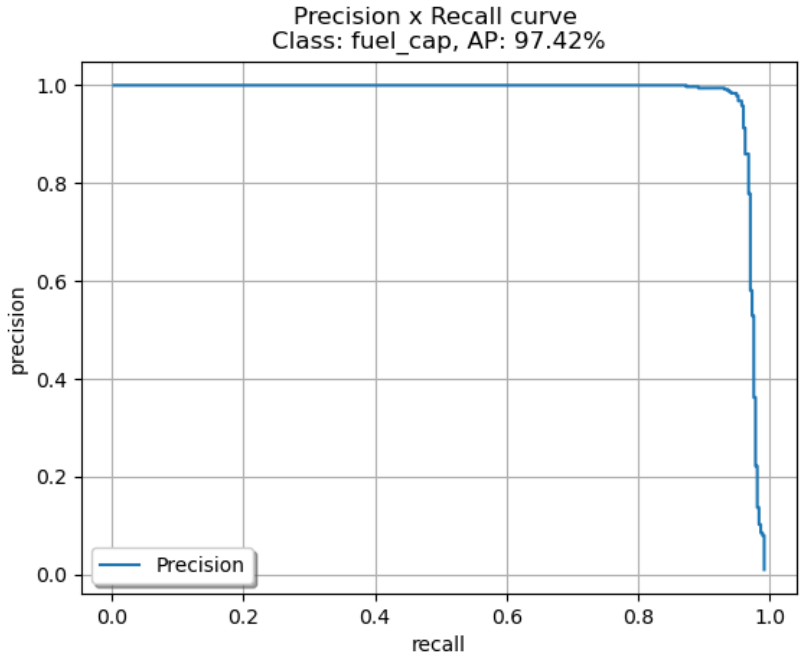


Figure 5.17: Precision x Recall curves for the better test of fuel cap object with hybrid images

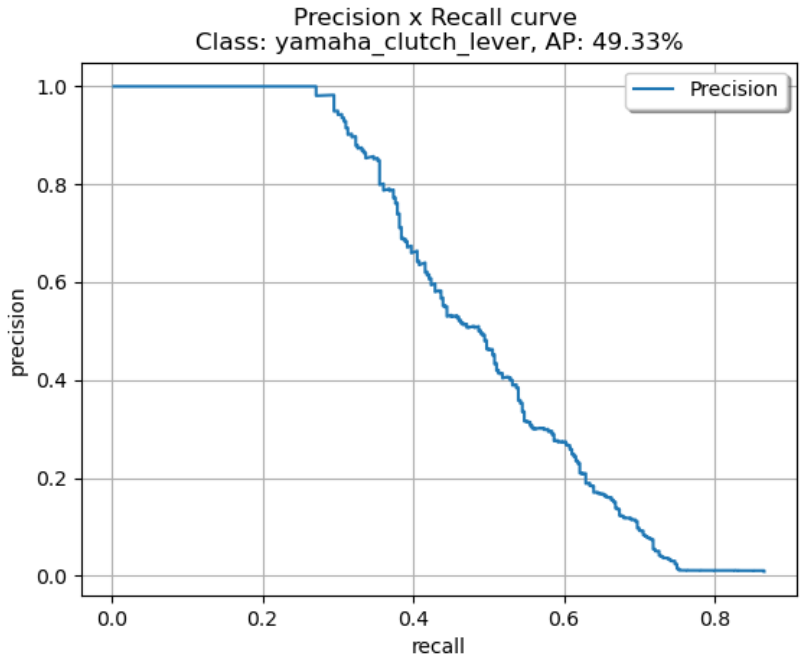


Figure 5.18: Precision x Recall curves for the better test of clutch lever object with hybrid images

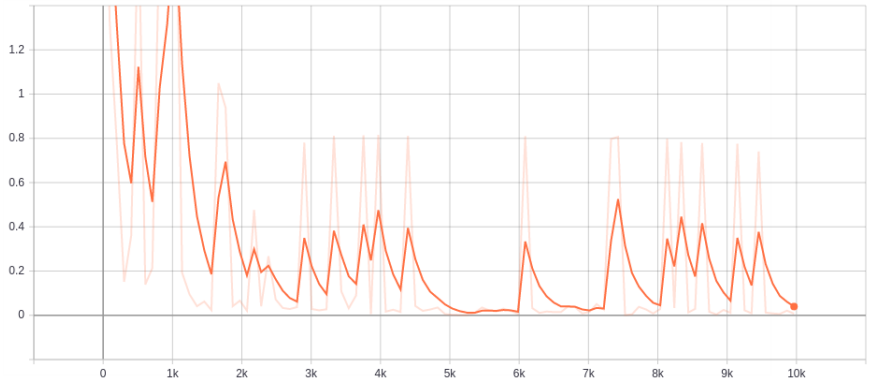


Figure 5.19: Loss function for better result of adblue object with hybrid images

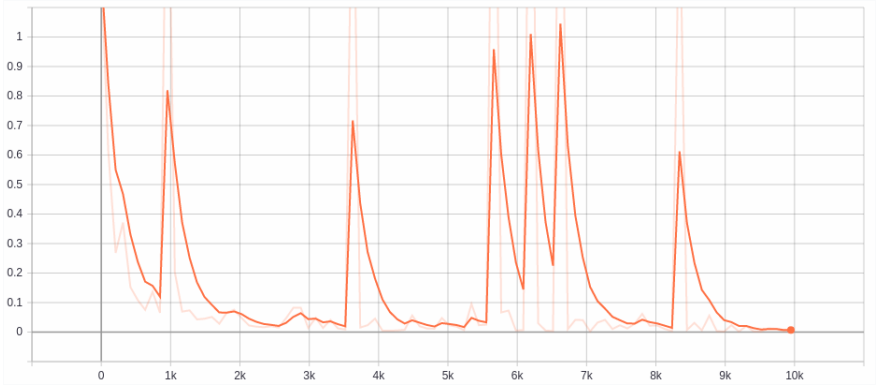


Figure 5.20: Loss function for better result of yamaha logo object with hybrid images

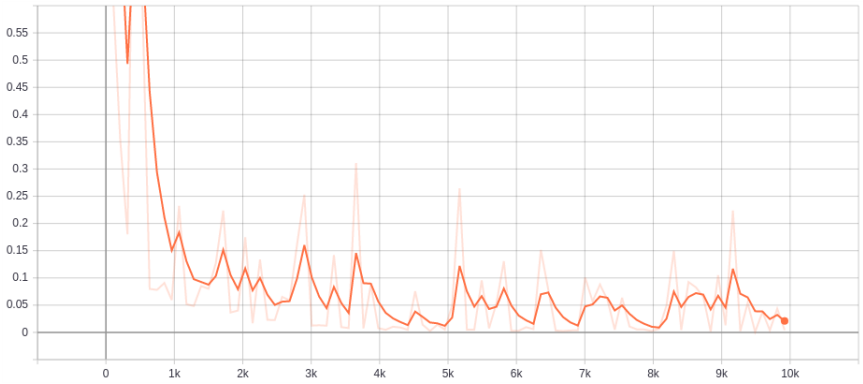


Figure 5.21: Loss function for better result of volkswagen logo object with hybrid images

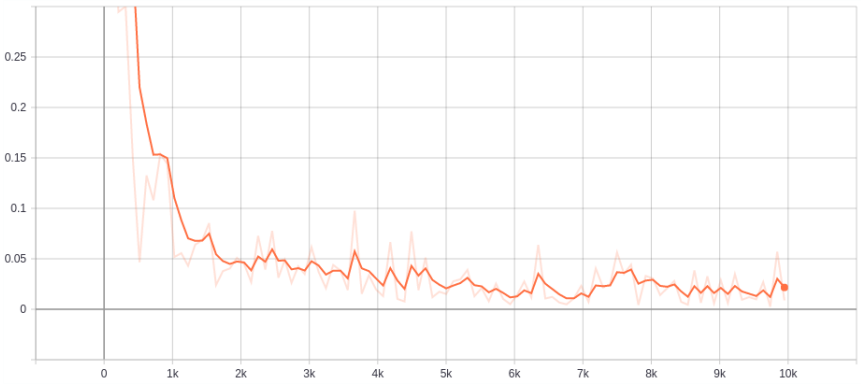


Figure 5.22: Loss function for better result of fuelcap object with hybrid images

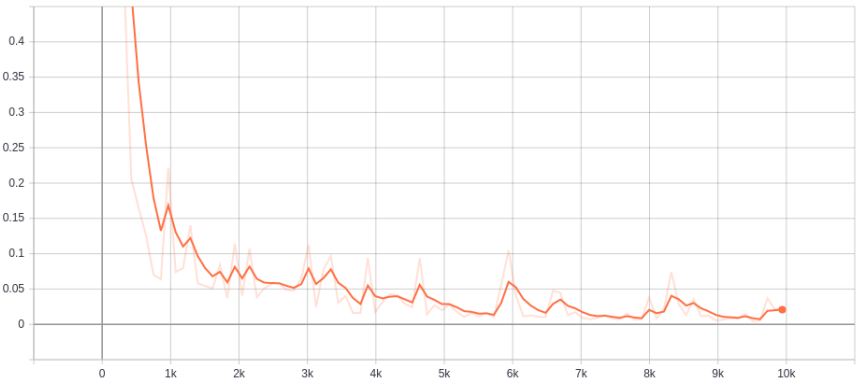


Figure 5.23: Loss function for better result of clutch lever object with hybrid images

5.3.3 Real Images Tests

With 100 real images and manually labeled images, 10 test samples were performed to obtain the results described in this section, not including the intermediate experiments, with other parameters and reduced number of samples.

The average and standard deviation of the metrics (precision, recall and F1-score) for each object can be seen in the Tables 5.12 and 5.13. The confusion matrices for the best results for each object can be seen in the Table 5.14. The corresponding Average Precision values [19], were also calculated and are reported in Precision x Recall curves for the IoU ≥ 0.5. The curves for the best results for each object can be seen in Figures 5.24, 5.25, 5.26, 5.27 and 5.28. The curves were plotted with python, using the algorithm developed in [39]. In Figures 5.29, 5.30, 5.31, 5.32 and 5.33, we can observe the loss functions for the best results for each object. The graphs of the loss function were plotted using the TensorBoard - TensorFlow API. The y axis represents the loss and the x axis represents the number of steps. The goal is to minimize the loss function. In other words, it means that the model is making fewer mistakes.

Object	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Adblue	89.66	87.33	89.57
Yamaha logo	30.00	0.22	0.44
Volkswagen logo	83.01	89.33	83.56
Fuel cap	84.36	36.18	49.54
Clutch lever	100.00	32.32	48.66

Table 5.12: Average of test samples for each object with real images

Object	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Adblue	7.48	4.28	4.83
Yamaha logo	48.30	0.47	0.92
Volkswagen logo	12.68	19.40	12.12
Fuel cap	11.46	10.42	9.77
Clutch lever	0.00	4.95	5.51

Table 5.13: Standard deviation of test samples for each object with real images

Object	contains	not contains
	adblue	not adblue
adblue	222	18
not adblue	13	-
	yamaha logo	not yamaha logo
yamaha logo	1	0
not yamaha logo	269	-
	volkswagen logo	not volkswagen logo
volkswagen logo	201	42
not volkswagen logo	0	-
	fuel cap	not fuel cap
fuel cap	212	43
not fuel cap	168	-
	clutch lever	not clutch lever
clutch lever	170	0
not clutch lever	210	-

Table 5.14: Confusion matrix for better results of each object with real images

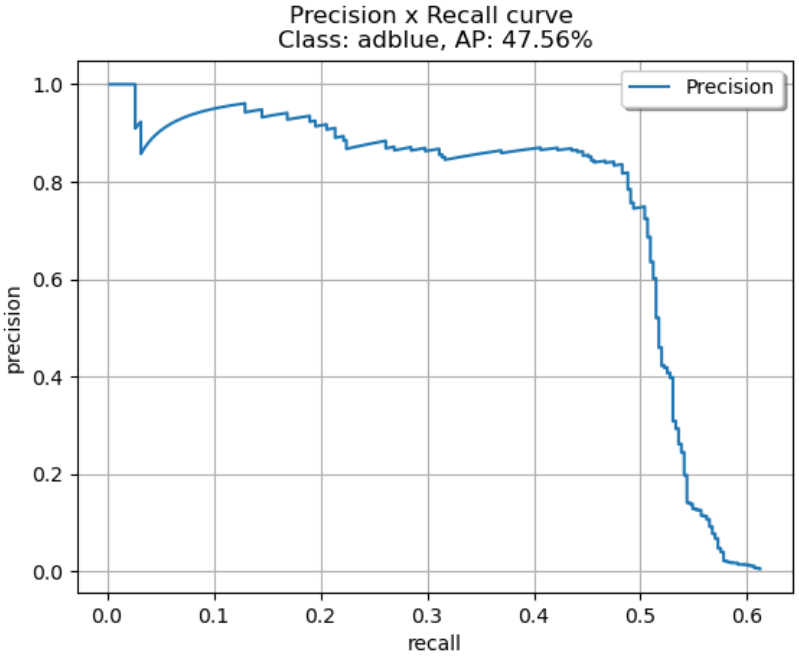


Figure 5.24: Precision x Recall curves for the better test of adblue object with real images

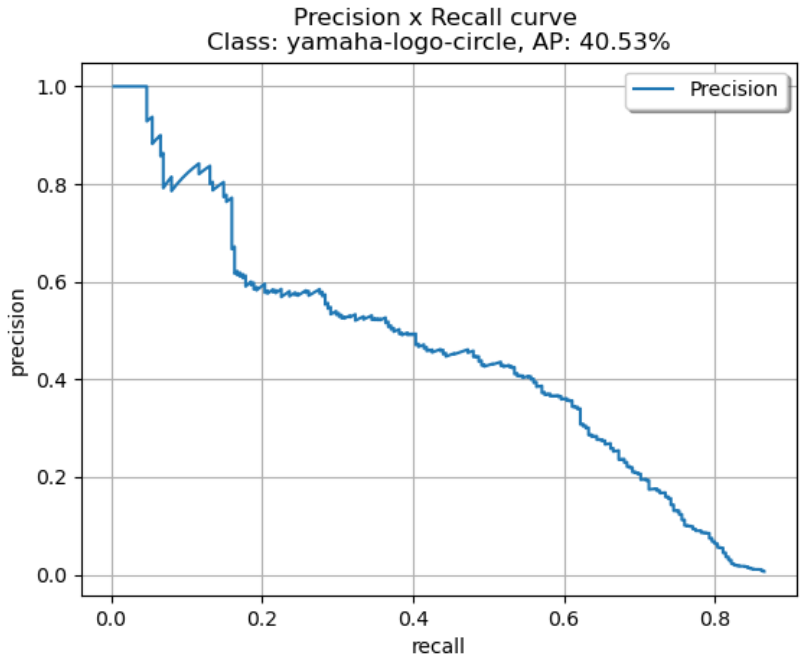


Figure 5.25: Precision x Recall curves for the better test of yamaha_logo object with real images

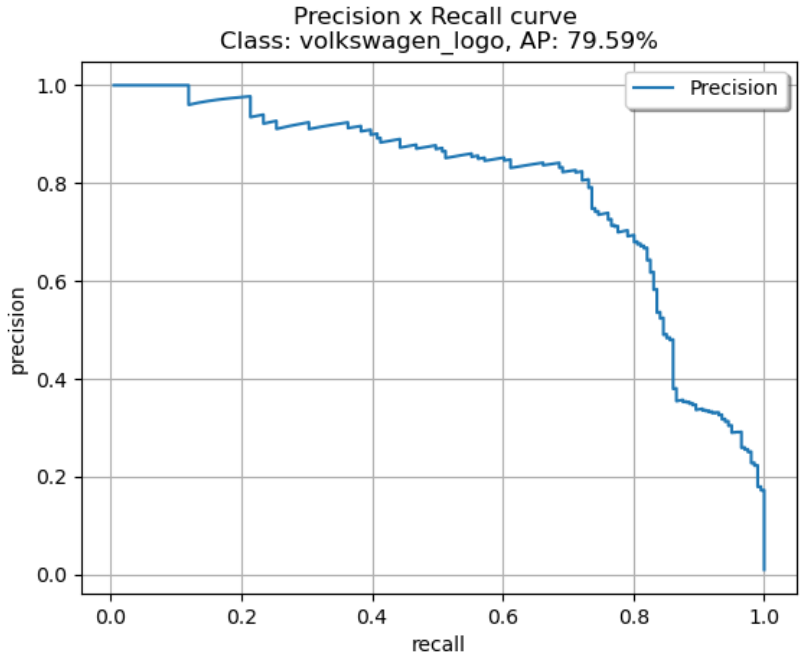


Figure 5.26: Precision x Recall curves for the better test of volkswagen_logo object with real images

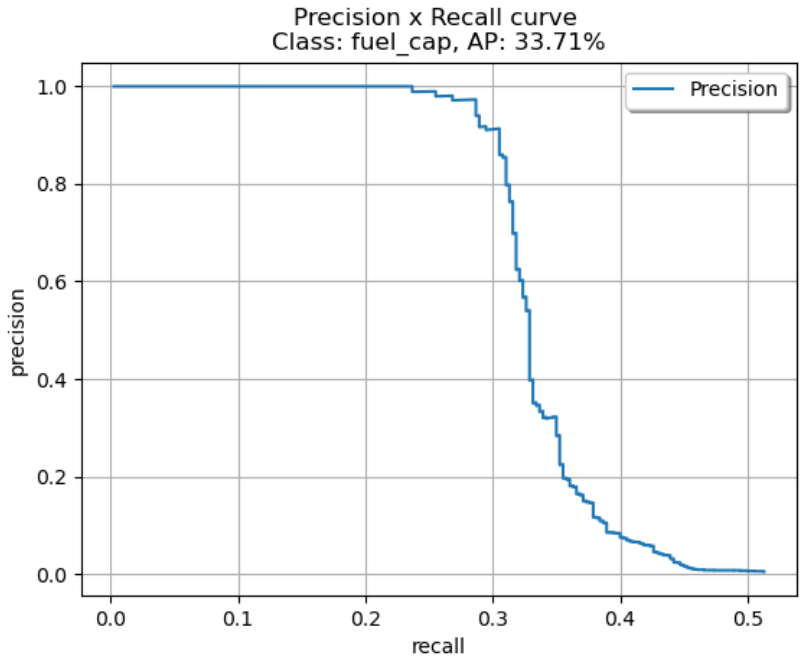


Figure 5.27: Precision x Recall curves for the better test of fuel cap object with real images

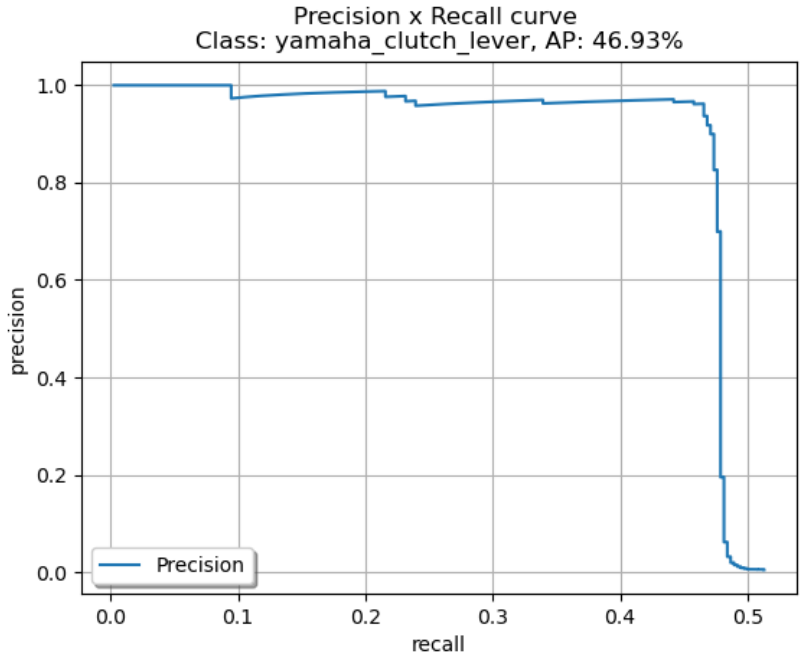


Figure 5.28: Precision x Recall curves for the better test of clutch lever object with real images

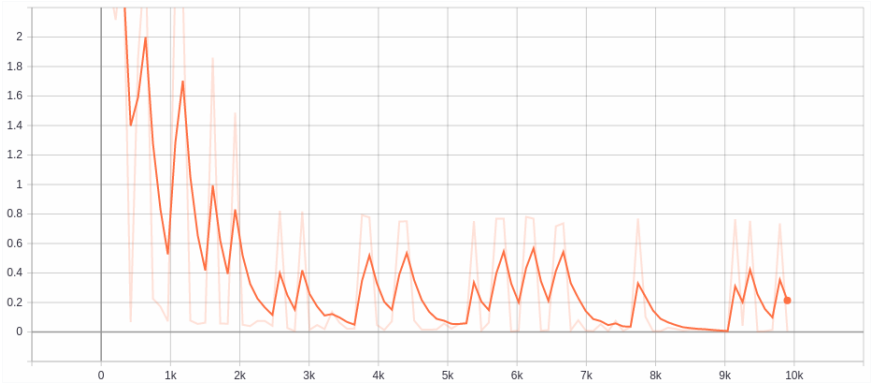


Figure 5.29: Loss function for better result of adblue object with real images

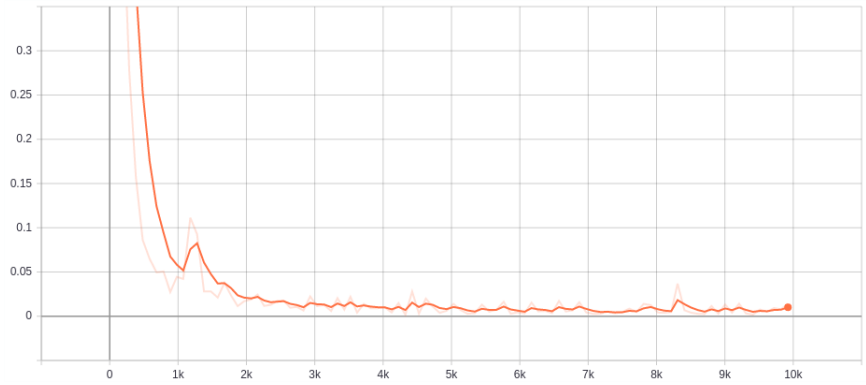


Figure 5.30: Loss function for better result of yamaha logo object with real images

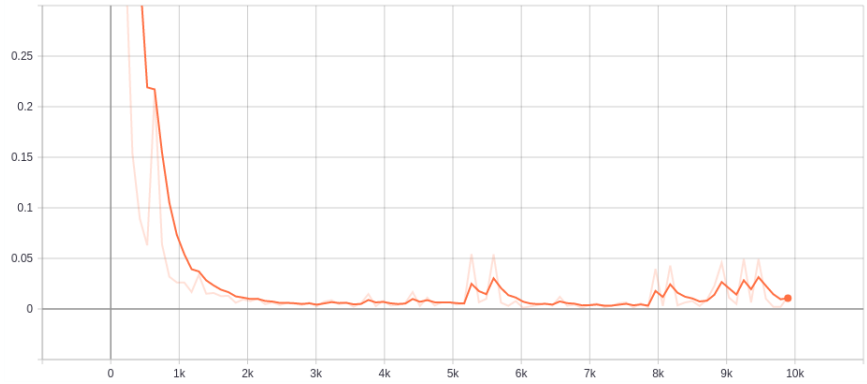


Figure 5.31: Loss function for better result of volkswagen logo object with real images

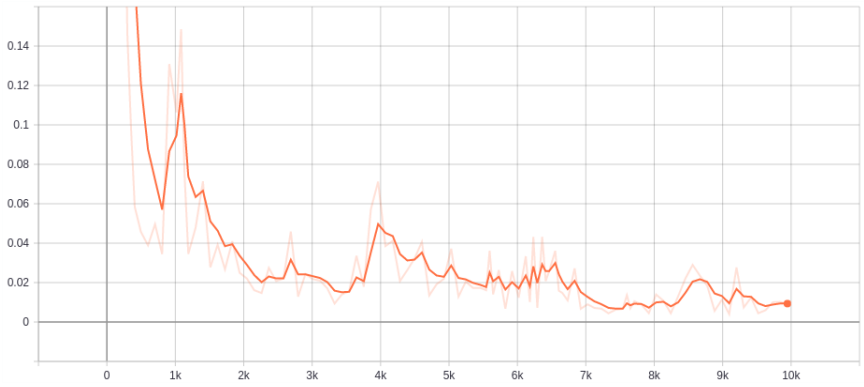


Figure 5.32: Loss function for better result of fuelcap object with real images

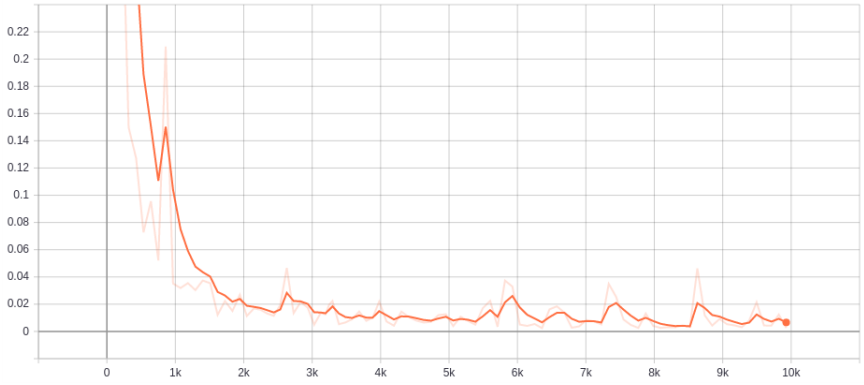


Figure 5.33: Loss function for better result of clutch lever object with real images

5.4 Discussion

It is difficult to compare our results with other works in the literature for some reasons. As far as we know, the approach presented in this work is the first proposal for an object detection model based on synthetic (automatically generated with a CAD model) and hybrid (synthetic + real) images applied to the industry.

Most OD works train and evaluate their synthetic image models in real datasets of common objects (cat, dog, train, plane, car, etc.), in order to validate not only the performance of the algorithm, but also setting hyperparameters. However, none of them is concerned with the development of an end-to-end systematization in industrial processes.

Despite this, in [36] the classification of objects in an automobile production line was tried, however, with only real images of the objects. In this work, the estimated detection accuracy was greater than 90 %. However, in our work, we have also achieved values above 90 % in several cases.

However, to give a general idea of the quality of our systematic approach, we can compare the results obtained with those reported in some studies. More generic works (with respect to objects), for example, in [46], for the detection of cats, dogs or trains, they obtained values of 89.4 %, 87.5 % and 82.1 % respectively. In [8], for the same objects, the values are 16 %, 5 % and 34 %.

Seeing these results, we can say that our approach produces good results in real images, being able to compete with the results obtained by the best OD models in standard datasets.

Finally, we can also compare our results with [21], in which the authors also train an OD model in synthetic images and evaluate in real images. However, this approach is not entirely systematic since the scenes are created manually in Blender. The object used in this work for evaluation is a glass of wine and the maximum AP obtained is 71.14 %. We can see that our systematic approach seems to work better than this approach, however, we cannot reproduce the method on our objects, as we cannot create the scenes manually in the same way that they would.

5.5 Hypothesis Tests

An alternative hypothesis H_{0_1}: $m_h > m_r$ was established (*"Is object detection using hybrid datasets better than using only real images?"*), in which m_h and m_r correspond to the higher precision values obtained (N=10) by the implemented models with hybrid images and real images, respectively. This alternative was tested for each object in question.

An another alternative hypothesis H_{0_2}: $m_s > m_r$ was established (*"Is object detection using synthetic images better than using real images?"*), in which m_s and m_r correspond to the higher precision values obtained (N=10) by the implemented models with synthetic images and real images, respectively. This alternative was tested for each object in question.

The Wilcoxon test [66] was chosen because it is nonparametric and appropriate for data from repeated-measures with two conditions. The test was performed on top of the three metrics (precision, recall, F1-score), for all test samples of all objects.

Object	Method	Mean	Std. Dev	SE Mean	p-value	Result	Parameter
Adblue	Hybrid	87.13	5.17	1.63	0.807	Probably the same distributions	Precision
	Real	89.66	7.48	2.37			Precision
Yamaha logo	Hybrid	95.04	3.79	1.20	0.973	Probably different distributions	Precision
	Real	30.00	48.30	15.28			Precision
Volkswagen logo	Hybrid	98.27	3.82	1.21	0.98	Probably different distributions	Precision
	Real	83.01	12.68	4.01			Precision
Fuel cap	Hybrid	94.22	2.86	0.90	0.973	Probably different distributions	Precision
	Real	84.36	11.46	3.62			Precision
Clutch Lever	Hybrid	73.38	21.73	6.87	0.998	Probably different distributions	Precision
	Real	100	0	0			Precision
Adblue	Hybrid	44.64	7.26	2.29	0.998	Probably different distributions	Recall
	Real	87.83	4.28	1.35			Recall
Yamaha logo	Hybrid	95.79	4.33	1.37	0.998	Probably different distributions	Recall
	Real	0.22	0.47	0.15			Recall
Volkswagen logo	Hybrid	62.98	18.61	5.89	0.98	Probably different distributions	Recall
	Real	89.33	19.4	6.14			Recall
Fuel cap	Hybrid	91.73	5.94	1.88	0.998	Probably different distributions	Recall
	Real	36.19	10.42	3.30			Recall
Clutch Lever	Hybrid	29.11	4.03	1.27	0.725	Probably the same distributions	Recall
	Real	32.32	4.95	1.57			Recall
Adblue	Hybrid	58.70	6.30	1.99	0.998	Probably different distributions	F1-Score
	Real	88.57	4.83	1.53			F1-Score
Yamaha logo	Hybrid	95.31	2.65	0.84	0.998	Probably different distributions	F1-Score
	Real	0.44	0.92	0.29			F1-Score
Volkswagen logo	Hybrid	75.06	13.36	4.22	0.916	Probably the same distributions	F1-Score
	Real	8.55	12.13	3.83			F1-Score
Fuel cap	Hybrid	92.84	3.38	1.07	0.998	Probably different distributions	F1-Score
	Real	49.54	9.77	3.09			F1-Score
Clutch Lever	Hybrid	41.17	7.37	2.33	0.895	Probably the same distributions	F1-Score
	Real	48.661	5.52	1.74			F1-Score

Table 5.15: The Wilcoxon test for the models considering the hybrid and real objects.

Object	Method	Mean	Std. Dev	SE Mean	p-value	Result	Parameter
Adblue	Synthetic	85.11	10.49	3.32	0.626	Probably the same distributions	Precision
	Real	89.66	7.48	2.37			Precision
Yamaha logo	Synthetic	78.03	16.30	5.15	0.973	Probably different distributions	Precision
	Real	30.00	48.30	15.28			Precision
Volkswagen logo	Synthetic	-	-	-	-	-	Precision
	Real	-	-	-			Precision
Fuel cap	Synthetic	-	-	-	-	-	Precision
	Real	-	-	-			Precision
Clutch Lever	Synthetic	-	-	-	-	-	Precision
	Real	-	-	-			Precision
Adblue	Synthetic	80.00	4.46	1.41	0.994	Probably different distributions	Recall
	Real	87.83	4.28	1.35			Recall
Yamaha logo	Synthetic	96.23	4.29	1.36	0.998	Probably different distributions	Recall
	Real	0.22	0.47	0.15			Recall
Volkswagen logo	Synthetic	-	-	-	-	-	Recall
	Real	-	-	-			Recall
Fuel cap	Synthetic	-	-	-	-	-	Recall
	Real	-	-	-			Recall
Clutch Lever	Synthetic	-	-	-	-	-	Recall
	Real	-	-	-			Recall
Adblue	Synthetic	81.93	3.88	1.23	0.98	Probably different distributions	F1-Score
	Real	88.57	4.83	1.53			F1-Score
Yamaha logo	Synthetic	85.19	11.15	3.52	0.998	Probably different distributions	F1-Score
	Real	0.44	0.92	0.29			F1-Score
Volkswagen logo	Synthetic	-	-	-	-	-	F1-Score
	Real	-	-	-			F1-Score
Fuel cap	Synthetic	-	-	-	-	-	F1-Score
	Real	-	-	-			F1-Score
Clutch Lever	Synthetic	-	-	-	-	-	F1-Score
	Real	-	-	-			F1-Score

Table 5.16: The Wilcoxon test for the models considering the synthetic and real objects.

In both tests (H_{0_1} e H_{0_2}), the p-value was chosen as 5 % of significance level. In Table 5.15 and 5.16 it is possible to view the results from the statistical test for each scenario. For some cases, there are significant differences between the performance of the models.

For the precision metric, of the 5 objects evaluated, 3 of them (yamaha logo, volkswagen logo and fuel cap) were better for the comparative case *hybrid vs real* and 1 of them (yamaha logo) was better for the comparative synthetic case vs real. Still regarding this metric, for the case of adblue object, the test *hybrid vs real* was not conclusive because the cases probably followed the same distribution. The *synthetic vs real* test for the same object (adblue) was also not conclusive, probably indicating the same data distribution. For the volkswagen logo, fuel cap and clutch lever objects in the *synthetic vs real* test, the synthetics cases of these objects were overfitted, making it impossible to make a concrete

assessment.

As for the recall metric, of the 5 objects evaluated, 2 of them (yamaha logo and fuel cap) were better for the comparative case *hybrid vs real* and 1 of them (yamaha logo) was better for the comparative case *synthetic vs real*. Still in relation to this metric, for the case of the clutch lever object, the test *hybrid vs real* was not conclusive because the cases probably followed the same distribution. For the volkswagen logo, fuel cap and clutch lever objects in the *synthetic vs real* test, the synthetics cases of these objects were overfitted, in this case, synthetic did not work for these objects, so distributions are definitely different.

Finally, for the f1-score metric, of the 5 objects evaluated, 2 of them (yamaha logo, and fuel cap) were better for the comparative case *hybrid vs real* and 1 of them (yamaha logo) was better for the case comparative *synthetic vs real*. Still regarding this metric, for the cases of volkswagen logo object and clutch lever object, the test *hybrid vs real* was not conclusive because the cases probably followed the same distribution. For the volkswagen logo, fuel cap and clutch lever objects in the *synthetic vs real* test, the synthetics cases of these objects were overfitted, making it impossible to make a concrete assessment.

Therefore, the use of hybrid models is recommended, because only synthetic is risky (risk of overfitting) and only real too.

Chapter 6

Conclusion

This master's thesis studies the problem of object detection in the industrial scenario using synthetic images, trying to understand which aspects of the process influence the final performance of the detector and proposing a systematic approach. The aspects consisted of the synthetic image generation process with labels, the training process and the inference of a given data set. A data set was also created for this work, with examples of labeled industrial images. The training process was carried out using a neural network, which sought to detect the object inserted in the image.

Detecting an object or multiple objects is not an extremely complex task. However, when it comes to the use of synthetic images in an industrial environment, this task becomes complex, as the training is done using only the CAD model of the object in question, in addition to the environment being extremely varied (objects, pieces and scenery, etc.). In addition, there is also a difficulty in finding CAD models of specific industrial objects so that they can be trained. Thus, as a consequence of this work, a set of data was produced and made publicly available for other research [1].

Faster RCNN was the neural network chosen to incorporate the method because it was the one that had the best time vs. precision relationship [2] among the models available in the TensorFlow API. The image rendering software, Blender, was chosen because it has a highly efficient Python API for automatic image generations, besides being very well documented.

At the end of the experiments, it was possible to conclude that the systematic approach developed in this work to detect objects in industrial environments from synthetic

[1] `https://github.com/igorgbs/systematic\_approach\_cad\_models`

[2] `https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md`

images, served to accurately detect their presence in images in industrial environments. Synthetic, real and hybrid images data sets were also tested statistically, where the latter was the one that obtained the best result. However, even using different objects, overfitting was a feature present in some cases. Even so, the results of the experiments were a success, enabling a discussion on the different possible approaches for detecting objects in industrial environments, introducing new ideas that can be implemented. In addition, it has been shown that a large set of images is not necessary to obtain a significant result. Our experiments indicate that the proposed rendering process is sufficient to obtain good performances and that the form of construction and rendering of the scenes is fundamental to the final result.

It is important to emphasize that the systematic approach developed in this work can be expanded to the most diverse types of CAD models of industrial parts. The main advantage of this approach is that anyone interested can use it to infer adjustments in the generation and training of the image, even before the real object begins to be produced. Another advantage is that this methodology avoids data collection and labeling errors.

6.1 Limitations

Some limitations were present in this work. The use of few objects was due to the fact that there are very few industrial and real CAD models for free, i.e. parts actually designed by the manufacturers and made available free of charge. As a result, it is also difficult to find image data sets with labeled CAD objects. In addition, it was not possible to implement the systematic approach directly in an assembly line. As for neural networks, it was also not possible to test with more Tensorflow models because there are about 40 models available. Textures and distractors truly found in a real production line were also not possible for the same reason as main objects. Another point of extreme importance concerns the inconclusive result of why overfitting occurs for some objects.

6.2 Future Work

There are several improvements that can be made. One of them would be to expand the systematic approach to more domains, being able to provide the construction of the datasets themselves, as was done in this work. Thus solving other computer vision problems. Additionally, obtain automatic textures and colors for objects present in real

images in order to try to avoid overfitting.

In addition, transforming the systematic approach into a framework or application could be one way. This could greatly facilitate industrial quality processes that involve detection and recognition of defective objects, for example.

With the use of sensors and networked machines it has resulted in the continuous generation of high volume data, another point of extreme importance would be the development of an appropriate architecture that would support object detection and recognition systems in industry 4.0.

References

[1] Baimukashev, D.; Zhilisbayev, A.; Kuzdeuov, A.; Oleinikov, A.; Fadeyev, D.; Makhataeva, Z.; Varol, H. A. Deep learning based object recognition using physically-realistic synthetic depth scenes. *Machine Learning and Knowledge Extraction 1*, 3 (2019), 883–903.

[2] Bardenet, R.; Brendel, M.; Kégl, B.; Sebag, M. Collaborative hyperparameter tuning. In *International conference on machine learning* (2013), pp. 199–207.

[3] Ben-Himane, S.; Hintestroisser, S.; Navab, N. Computer vision cad models, Oct. 14 2010. US Patent App. 12/682,199.

[4] Cheng, G.-J.; Liu, L.-T.; Qiang, X.-J.; Liu, Y. Industry 4.0 development and application of intelligent manufacturing. In *2016 international conference on information system and artificial intelligence (ISAI)* (2016), IEEE, pp. 407–410.

[5] Collins, P. K.; Leen, R.; Gibson, I. Industry case study: rapid prototype of mountain bike frame section. *Virtual and Physical Prototyping 11*, 4 (2016), 295–303.

[6] Community, B. O. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

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

[8] Dean, T.; Ruzon, M. A.; Segal, M.; Shlens, J.; Vijayanarasimhan, S.; Yagnik, J. Fast, accurate detection of 100,000 object classes on a single machine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 1814–1821.

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

[10] Dopico, M.; Gomez, A.; De la Fuente, D.; García, N.; Rosillo, R.; Puche, J. A vision of industry 4.0 from an artificial intelligence point of view. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)* (2016), The Steering Committee of The World Congress in Computer Science, Computer ..., p. 407.

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

[12] Everingham, M.; Eslami, S. A.; Van Gool, L.; Williams, C. K.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *International journal of computer vision 111*, 1 (2015), 98–136.

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

[14] Felzenszwalb, P.; McAllester, D.; Ramanan, D. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on computer vision and pattern recognition* (2008), IEEE, pp. 1–8.

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

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

[17] Hinterstoisser, S.; Lepetit, V.; Wohlhart, P.; Konolige, K. On pre-trained image features and synthetic images for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 0–0.

[18] Hirz, M.; Rossbacher, P.; Gulanová, J. Future trends in cad–from the perspective of automotive industry. *Computer-aided design and applications 14*, 6 (2017), 734–741.

[19] Hoiem, D.; Divvala, S. K.; Hays, J. H. Pascal voc 2008 challenge. In *PASCAL challenge workshop in ECCV* (2009), Citeseer.

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

[21] Jabbar, A.; Farrawell, L.; Fountain, J.; Chalup, S. K. Training deep neural networks for detecting drinking glasses using synthetic images. In *International Conference on Neural Information Processing* (2017), Springer, pp. 354–363.

[22] Jaccard, P. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat 37* (1901), 547–579.

[23] Jalled, F.; Voronkov, I. Object detection using image processing. *arXiv preprint arXiv:1611.07791* (2016).

[24] Kamencay, P.; Benčo, M.; Mižduš, T.; Radil, R. A new method for face recognition using convolutional neural network.

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

[26] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[27] Lafreniere, B.; Grossman, T. Blocks-to-cad: A cross-application bridge from minecraft to 3d modeling. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (2018), pp. 637–648.

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

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

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

[31] Lindsay, A.; Paterson, A.; GRAHAM, I. Identifying and quantifying inefficiencies within industrial parametric cad models. In *Advances in Manufacturing Technology XXXII: Proceedings of the 16th International Conference on Manufacturing Research, incorporating the 33rd National Conference on Manufacturing Research, September 11–13, 2018, University of Skövde, Sweden* (2018), vol. 8, IOS Press, p. 227.

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

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

[34] Luan, S.; Li, Y.; Wang, X.; Zhang, B. Object detection and tracking benchmark in industry based on improved correlation filter. *Multimedia Tools and Applications 77*, 22 (2018), 29919–29932.

[35] Mahalingam, T.; Subramoniam, M. A robust single and multiple moving object detection, tracking and classification. *Applied Computing and Informatics* (2020).

[36] Mazzetto, M.; Southier, L. F.; Teixeira, M.; Casanova, D. Automatic classification of multiple objects in automotive assembly line. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (2019), IEEE, pp. 363–369.

[37] Mitash, C.; Bekris, K. E.; Boularias, A. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), IEEE, pp. 545–551.

[38] Oh, Y.; Ransikarbum, K.; Busogi, M.; Kwon, D.; Kim, N. Adaptive svm-based real-time quality assessment for primer-sealer dispensing process of sunroof assembly line. *Reliability Engineering & System Safety 184* (2019), 202–212.

[39] Padilla, R.; Netto, S. L.; da Silva, E. A. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)* (2020), IEEE, pp. 237–242.

[40] Park, K.; Patten, T.; Prankl, J.; Vincze, M. Multi-task template matching for object detection, segmentation and pose estimation using depth images. In *2019 International Conference on Robotics and Automation (ICRA)* (2019), IEEE, pp. 7207–7213.

[41] Peng, X.; Sun, B.; Ali, K.; Saenko, K. Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1278–1286.

[42] Pinto, N.; Barhomi, Y.; Cox, D. D.; DiCarlo, J. J. Comparing state-of-the-art visual features on invariant object recognition tasks. In *2011 IEEE workshop on Applications of computer vision (WACV)* (2011), IEEE, pp. 463–470.

[43] Prasad, D. K. Survey of the problem of object detection in real images. *International Journal of Image Processing (IJIP) 6*, 6 (2012), 441.

[44] Rajpura, P. S.; Bojinov, H.; Hegde, R. S. Object detection using deep cnns trained on synthetic images. *arXiv preprint arXiv:1706.06782* (2017).

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

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

[47] Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (2015), pp. 91–99.

[48] Rozantsev, A.; Lepetit, V.; Fua, P. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding 137* (2015), 24–37.

[49] Ruiz, F. E.; Pérez, P. S.; Bonev, B. I. *Information theory in computer vision and pattern recognition*. Springer Science & Business Media, 2009.

[50] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision 115*, 3 (2015), 211–252.

[51] Sabokrou, M.; Fayyaz, M.; Fathy, M.; Moayed, Z.; Klette, R. Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. *Computer Vision and Image Understanding 172* (2018), 88–97.

[52] Salau, A. O.; Jain, S. Feature extraction: A survey of the types, techniques, applications. In *2019 International Conference on Signal Processing and Communication (ICSC)* (2019), IEEE, pp. 158–164.

[53] Sharma, K. U.; Thakur, N. V. A review and an approach for object detection in images. *International Journal of Computational Vision and Robotics 7*, 1-2 (2017), 196–237.

[54] Shirley, P.; Morley, R. K. *Realistic ray tracing.* AK Peters/CRC Press, 2003.

[55] Shleymovich, M.; Medvedev, M.; Lyasheva, S. Object detection in the images in industrial process control systems based on salient points of wavelet transform analysis. In *2016 2nd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)* (2016), IEEE, pp. 1–6.

[56] Shrestha, A.; Karki, N.; Yonjan, R.; Subedi, M.; Phuyal, S. Automatic object detection and separation for industrial process automation. In *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)* (2020), IEEE, pp. 1–5.

[57] Su, H.; Qi, C. R.; Li, Y.; Guibas, L. J. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 2686–2694.

[58] Sultana, F.; Sufian, A.; Dutta, P. A review of object detection models based on convolutional neural network. In *Intelligent Computing: Image Processing Based Applications*. Springer, 2020, pp. 1–16.

[59] Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence* (2017).

[60] Tremblay, J.; Prakash, A.; Acuna, D.; Brophy, M.; Jampani, V.; Anil, C.; To, T.; Cameracci, E.; Boochoon, S.; Birchfield, S. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2018), pp. 969–977.

[61] Tsai, C.; Lin, W. A comparative study of global and local feature representations in image database categorization. In *2009 Fifth International Joint Conference on INC, IMS and IDC* (2009), pp. 1563–1566.

[62] Van de Sande, K. E.; Uijlings, J. R.; Gevers, T.; Smeulders, A. W. Segmentation as selective search for object recognition. In *2011 International Conference on Computer Vision* (2011), IEEE, pp. 1879–1886.

[63] Viola, P.; Jones, M. 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* (2001), vol. 1, IEEE, pp. I–I.

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

[65] Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience 2018* (2018).

[66] Wilcoxon, F. Individual comparisons by ranking methods. biometrics bulletin 1, 6 (1945), 80–83. *URL http://www. jstor. org/stable/3001968* (1945).

[67] Xiao, Y.; Tian, Z.; Yu, J.; Zhang, Y.; Liu, S.; Du, S.; Lan, X. A review of object detection based on deep learning. *Multimedia Tools and Applications* (2020), 1–63.

[68] Xu, L. D.; Xu, E. L.; Li, L. Industry 4.0: state of the art and future trends. *International Journal of Production Research 56*, 8 (2018), 2941–2962.

[69] Yang, J.; Li, S.; Wang, Z.; Yang, G. Real-time tiny part defect detection system in manufacturing using deep learning. *IEEE Access 7* (2019), 89278–89291.

[70] Zhang, X.; Yang, Y.-H.; Han, Z.; Wang, H.; Gao, C. Object class detection: A survey. *ACM Computing Surveys (CSUR) 46*, 1 (2013), 1–53.

[71] Zhang, Z.; Li, J.; Shao, W.; Peng, Z.; Zhang, R.; Wang, X.; Luo, P. Differentiable learning-to-group channels via groupable convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 3542–3551.

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

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