KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Technology

Exploiting scene constraints to improve object detection algorithms for industrial applications

ir. Puttemans Steven

Supervisors:
Prof. dr. ir. T. Goedemé
Prof. dr. ir. T. Tuytelaars

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Technology (PhD)

December 2017

Exploiting scene constraints to improve object detection algorithms for industrial applications

ir. Puttemans STEVEN

Examination committee:
Prof. dr. ir. M. Versteyhe, chair
Prof. dr. ir. T. Goedemé, supervisor
Prof. dr. ir. T. Tuytelaars, co-supervisor
Prof. dr. J. Vennekens
Prof. dr. ir. L. Van Eycken
ir. R. den Boer
Dr. M. Proesmans
(KU Leuven)
Prof. dr. ir. E. Gavves
(University of Amsterdam (UVA))

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Technology (PhD)

December 2017

Preface

Looking back at the last 5 years, I could have never imagined that pursuing a PhD would be such an adventure. Attending various conferences, presenting and discussing my latest achievements, etc. really improved my skill-set and definitely contributed to the quality of this dissertation. Meeting tons of like-minded academics and seeing them push the state-of-the-art in computer vision and artificial intelligence even further, it all has driven me to where I am today. Let us not forget the numerous exotic and sunny locations my research brought me like Tokyo, Lisbon, Rome, Barcelona, etc. and even the harsh cold winters of Finland. All memories made for life and things I probably would never have done if I had not pursued my doctoral degree. Tackling this adventure would have never been possible without the support of numerous people, which I would like to thank accordingly.

First of all, I want to thank my girlfriend, Lisa, for being there for me, believing in me and always supporting me in my decisions along the way, even if that often meant coping with a grumpy and tired boyfriend. You are and always will be the most important person in my life. Thank you for everything so far!

I would like to thank my supervisor, prof. dr. ir. Toon Goedemé, for offering me the opportunity of starting my doctoral research. Thank you for guiding me in the last six years, providing heaps of feedback on everything, from papers to presentations, always ensuring a more qualitative end result. You allowed me to make mistakes, to learn from them and guided me towards new insights, all of which surely contributed to my academic mindset. I sincerely hope we can continue our cooperation within EAVISE for the upcoming years!

I would also like to thank my co-supervisor prof. dr. ir. Tinne Tuytelaars, for being my supervisor during my first PhD year, as well as all members of my examination committee: prof. dr. Joost Vennekens, prof. dr. ir. Luc Van Eycken, dr. Marc Proesmans, ir. Rien den Boer and prof. dr. ir. Efstratios Gavves, for their valuable feedback along the way and their time spend proofreading this dissertation.

Of course, working at the EAVISE research group would not have been the same without my colleagues. The numerous coffee breaks and discussions, the extra activities like carting, playing pool and the team-buildings, all contributed to a nice work atmosphere. For this, I would like to thank each and every one of you. Special thanks to Kristof, Stijn and Dries, for creating great moments at conferences, for always helping me out when I needed advice or whenever I looked for someone to share some thoughts with.

Last but not least, I should definitely thank my parents. Mom, Dad, you both allowed me to make some mistakes during my studies, but always kept pushing me to reach further and aim higher. I am sure that without your support I would not have been where I am today! And of course, I cannot forget my family and family-in-law. Thank you all for always being interested in my progress and supporting me throughout these years.

The research in this dissertation would not have been possible without the help of several companies, providing the necessary data to this research: Aris BV, Vansteelandt BVBA, Grontmij Belgium, VITO, Research Center for Fruit and Infrax. Furthermore, we would like to explicitly thank the Flanders Innovation & Entrepreneurship (VLAIO) for the partial funding and NVIDIA for providing a Titan X GPU as hardware grant.

I would like to finish this preface with the following words, knowing they are the basis of what has brought me this far, hoping they will guide me in a bright future ahead.

> *"The only limit to the height of your achievements is the reach of your dreams and your willingness to work for them."*
>
> – Michelle Obama

Abstract

State-of-the-art object detection algorithms are designed to be heavily robust against scene and object variations like illumination changes, occlusions, scale changes, orientation differences, background clutter and object intra-class variability. This allows users to use these algorithms *in-the-wild*, at any given location under any given circumstances. However, in industrial machine vision applications, where objects with variable appearance have to be detected, many of these variations are in fact constant and can be seen as scene- and application-specific constraints on the detection problem.

This dissertation takes these constraints and investigates how these can be used to reduce the enormous search space for object candidates, speeding up the actual detection process while simultaneously increasing the achieved accuracy. While doing so we focus on three major aspects of object detection:

1. The amount of training data needed to obtain a robust and highly accurate detection model. We suggest using our scene- and application-specific constraints to reduce the amount of needed training data as much as possible and focus on those training samples that actually matter for training an effective object detection model. By doing so we reduce the amount of manual labour, needed for labelling these training samples, as much as possible. Furthermore, we introduce an innovative active learning approach that helps us narrow down the useful training samples even more while maintaining a high detection accuracy.

2. Maintaining real-time processing speeds. This is a hard-constraint in many industrial applications but several state-of-the-art object detection algorithms do not yet succeed in achieving these processing speeds. By smartly using the scene- and application-specific constraints we aim at increasing the speed of the detection process, and removing image areas that do not contain objects from the search process as fast as possible.

3. Reduce the amount of false positive *(non-object image regions detected as objects)* and false negative *(object image regions detected as non-object regions)* detections as much as possible, which helps at increasing the overall average precision of the object detector.

We work on these three aspects of object detection both at training and at inference time, simplifying both steps of the process. Since data is, according to literature, crucial in obtaining robust object detection models, we investigate how we can increase a small meaningful set of training data into more meaningful data using data augmentation.

To be able to work on all these aspects of object detection, by looking at scene- and application-specific constraints, we progressively increase the complexity of our approach. We first start by using the constraints as pre- or post-filtering operations on the output of a generic off-the-shelf object detector, which already allowed us to drastically remove the number of false positive detections and thus boosting the average precision of our models.

Proving their usefulness, we took the power of these constraints a bit further and integrated scene- and application-specific knowledge into the actual training process of the model. This not only allowed to reduce the number of false positive detections, it also allowed a significant drop in false negative detections. This again pushed the average precision of our models higher, compared to standard off-the-shelf detectors.

When we proved the power of using these constraints during both training and inference, we took a look at reducing the amount of manually labelled training data as much as possible. This is done by suggestion an innovative active learning approach that uses weak classifiers to look for meaningful and much-needed training samples, only presenting those to the human annotator. This not only increased the average precision, but also dropped the manual annotation from multiple ten thousands to only several hundreds of images.

At the end of our PhD, we investigated the possibilities of using deep learning as an object detection architecture. We proved that given the combination of data augmentation and transfer learning, it is possible to train application specific deep learning object detection models, which given a very limited set of training samples *(eg. for some models only 15 object specific samples were used)* is able to achieve average precisions of 99% and higher, creating optimal solutions for our case specific object detection tasks.

Finally, we integrated all this research into several industrially relevant applications and generated a lessons-learned chapter filled with practical tips and know-how on applying object detection to industrially relevant problems.

Beknopte samenvatting

State-of-the-art objectdetectie algoritmes worden ontworpen met een hoge robuustheid ten opzichte van scène- en object-variaties zoals een wijzigende belichting, occlusie, variatie in schaal en oriëntatie, achtergrondruis en intra-klasse variatie. Door deze variaties in de detector in te bouwen, kunnen we deze technieken in-het-wild gebruiken, op elke mogelijke locatie en onder elke mogelijke omstandigheid. Daartegenover staan industriële beeldverwerkingstoepassingen, waar objecten met een grote variatie voorkomen en gedetecteerd dienen te worden, maar waar veel van deze variaties in feite constant en gekend zijn. Hierdoor kunnen we ze gebruiken als scène- en applicatie-specifieke beperkingen op het objectdetectie probleem.

Deze doctoraatstekst bekijkt hoe deze beperkingen gebruikt kunnen worden om de gigantische zoekruimte van object-kandidaten te reduceren, waardoor het detectie-proces versneld wordt en tegelijk de accuraatheid omhoog gaat. Dit doen we telkens met drie belangrijke aspecten in het achterhoofd:

1. De hoeveelheid trainingsdata nodig om een robuust en accuraat objectdetectie model te bekomen. We suggereren om gebruik te maken van onze scène- en applicatie-specifieke beperkingen om de hoeveelheid trainingsdata zoveel mogelijk te reduceren en om te focussen op de trainingsvoorbeelden die effectief bijdragen aan het training van een robuuste objectdetector. Hierdoor reduceren we de hoeveelheid manuele input die nodig is voor het labelen van de trainingsvoorbeelden zoveel als mogelijk. Daarenboven introduceren we een innovatieve actieve learning strategie die ons helpt de belangrijke trainingsvoorbeelden uit te filteren en tegelijkertijd de hoge detectie accuraatheid te bewaren.
2. Bewaken van real-time uitvoer snelheden. Dit is een niet te negeren vereiste opgelegd door de eigenheid van industriële objectdetectie toepassingen, maar verscheidene state-of-the-art algoritmes halen deze realtime uitvoer snelheid nog niet. Door op een slimme manier onze scène- en applicatie-specifieke beperkingen te gebruiken, trachten we de snelheid van het detectieproces op te voeren en beeldregio's die geen objecten bevatten zo snel mogelijk uit het zoekproces te halen.

3. Zoveel mogelijk reduceren van vals positieve *(achtergrondregio's uit het beeld die gedetecteerd worden als object)* en vals negatieve *(object regio's die gedetecteerd worden als achtergrondregio's)* detecties, wat bijdraagt aan een toename in gemiddelde accuraatheid van de object detector.

Doorheen het doctoraatsonderzoek werken we aan deze aspecten zowel tijdens de training als tijdens de inferentie, met het oog op een vereenvoudiging van beide stappen. Aangezien data, volgens de literatuur, cruciaal is voor het trainen van een robuust detectiemodel, onderzoeken we hoe een kleine betekenisvolle set trainingsdata vergroot kan worden via data augmentatie.

Met onze scène- en applicatie-specifieke beperkingen in het achterhoofd, laten we de complexiteit van onze voorgestelde aanpak stelselmatig toenemen. We starten allereerst door de beperkingen om te gieten in pre- of post-filter operaties op de input of output van een off-the-shelf objectdetectie algoritme. Dit laat ons toe om de beperkingen te gebruiken om het aantal vals positieve detecties drastisch te doen dalen en de gemiddelde accuraatheid omhoog te duwen. Het gebruik van deze filters toont de bruikbaarheid van de beperkingen aan, maar we gaan nog een stap verder en integreren de scène- en applicatie-specifieke kennis rechtstreeks in het leerproces van het detectiemodel. Dit laat niet enkel toe om het aantal vals positieve detecties verder te doen dalen, het staat ons ook toe de hoeveelheid vals negatieve detecties te reduceren.

Na het aantonen van de bijdrage van deze specifieke kennis tijdens zowel de training als de inferentie, nemen we de hoeveelheid van manueel gelabelde trainingsvoorbeelden onder de loep, met het oog op het aantal zoveel mogelijk te doen dalen. Dit doen we door een innovatieve active learning strategie te implementeren die via zwakke detectoren op zoek gaat naar betekenisvolle trainingsvoorbeelden, die vervolgens aan de annotator worden voorgeschoteld. Dit vermindert de hoeveelheid manuele annotaties van verscheidene tienduizenden tot slechts enkele honderden afbeeldingen.

Op het eind van dit doctoraatsonderzoek onderzochten we de mogelijkheden van deep learning als objectdetectiearchitectuur. We tonen aan dat de combinatie van data augmentatie en transfer learning het mogelijk maakt om applicatie specifieke deep learning architecturen te gebruiken voor objectdetectie, wetende dat we slechts een beperkte hoeveelheid trainingsdata hebben *(bij sommige modellen gebruikten we slechts 15 objectspecifieke voorbeelden)*. We behalen een gemiddelde accuraatheid van 99%, waardoor een optimale oplossing voor onze specifieke industriële objectdetectie opdrachten zich opdringt.

Tot slot integreerden we al deze kennis in tal van industrieel relevante applicaties en schreven we een reeks praktische tips en know-how over toegepaste industriële objectdetectie neer in een *'lessons-learned'* hoofdstuk.

List of Abbreviations

ACF	Aggregated Channel Features. Pedestrian detection methodology, based on ICF, as described in [24].
AdaBoost	Adaptive Boosting. A machine learning algorithm for selecting strong features from a large feature pool, as described in [35].
AP	Average precision. The precision averaged over all values of the recall (i.e. identical to the AUC).
AUC	Area Under the Curve. Often indicates the area under a PR-curve *(percentual)*, and thus a measure of accuracy.
BSD	Licensing model that allows integrating the open-source software in closed-source applications without paying license costs.
CMYK	A colour space used frequently in printing, based on 4 colours: cyan, magenta, yellow and black.
CNN	Convolutional Neural Networks. A state-of-the-art deep learning methodology for performing tasks like object detection, classification and segmentation, as described in [55].
CUDA	A parallel computing platform and application programming interface (API) model created by NVIDIA.
Darknet19	A deeply learned architecture for object classification by Redmond et al containing 19 convolutional layers, mimicking the VGG19 model.
Densenet201	A deeply learned architecture for object classification by Redmond et al containing 201 convolutional layers, mimicking the behaviour of [47].

DoG Difference of Gaussians, a feature enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original, which suppresses high spatial frequencies in images.

DPM Deformable Part Models. An object detection methodology which employs a non-rigid template model, as described in [32].

FDDB Face Detection Data Set and Benchmark, a data set of face regions designed for studying the problem of unconstrained face detection.

FGIA Flanders Geographical Information Agency.

FN False Negative. Evaluation metric of an object detection algorithm, indicating instance that is incorrectly classified as not belonging to the class to be detected.

FPS Frames per second. Indicates the throughput (i.e. speed) of a camera or an object detection algorithm in the context of this dissertation.

FP False Positive. Evaluation metric of an object detection algorithm, indicating instance that is incorrectly classified as belonging to the class to be detected.

GB Gigabit, a metric for defining RAM memory size, indicating 1.073.741.824 bytes.

GIS Geographic Information System, a system designed to capture, store, manipulate, analyse, manage, and present spatial or geographic data.

GPGPU General Purpose Graphical Processing Unit. Dedicated hardware for doing fast and complex calculations.

GT Ground truth, indicating object label and location to the image dimensions.

HOG Histogram of Oriented Gradients. A technique for describing pixel intensity gradients in an image and their orientation, as described in [16].

HSV One of the most common cylindrical-coordinate representations of points in an RGB colour model.

ICF Integral Channel Features. A way of combining multiple feature channels into a single object detector, tested as pedestrian detection methodology, as described in [26].

InceptionV3 The third version of a deeply learned architecture by Google introducing the Inception module, as described in [110].

IoU Intersection over union. A metric used for matching ground truth annotations and found detections after non-maxima suppression. It is the area of overlap between both bounding boxes divided by the area of union of both bounding boxes.

LBP Local Binary Patterns. A technique for describing features in an image, based on intensity differences around the centre pixel, as described in [43].

LWIR Long Wave Infra-Red.

MB Megabit, a metric for defining RAM memory size, indicating 1.048.576 bytes.

MSER Maximally Stable Extremal Regions. An algorithm for extracting features in an RGB image.

NMS Non-maxima suppression. A technique to reduce the number of bounding boxes after a sliding window evaluation. In the case of overlapping bounding boxes (based on an overlap criterion) only the highest scoring bounding box is maintained.

OCR Optical Character Recognition. A technique to automatically capture written text.

OHTA A colour space introduced by Ohta et al in [78].

OpenCV An open-source computer vision library supplied by Intel, as described in [12].

PR-curve A curve which visualises the trade-off between the precision and the recall.

PR Precision-Recall, a universal metric for evaluation object detection algorithms. P = precision = fraction of the detected objects that are actual objects. R = recall = fraction of actual objects that are detected.

R-CNN Region Proposing Convolution Neural Networks. An object detection methodology which uses region proposals as input to a deep learning architecture, as described in [98].

RAM Random Access Memory. A form of computer data storage which stores frequently used program instructions to increase the general speed of a system.

RF Random Forests. A Machine learning algorithm for optimally separating two feature sets, as described in [36].

RGB Red, Green, Blue. The three colour channels of a colour image.

SME Small and medium-sized enterprise.

SSD Single Shot Multibox Detector. An algorithm for detecting object using only a single pass deep neural network, as described in [67].

SVM Support Vector Machine. A machine learning algorithm for optimally separating two feature sets, as described in [15].

TN True Negative. Evaluation metric of an object detection algorithm, indicating instance that is correctly classified as not belonging to the class to be detected.

TP True Positive. Evaluation metric of an object detection algorithm, indicating instance that is correctly classified as belonging to the class to be detected.

UAV Unmanned Aerial Vehicle. Commonly known as drone and seen as an aircraft without a human pilot.

VGA Video Graphics Array, indicating a resolution of 640×480.

VGG19 A deeply learned architecture for object classification by the Visual Geometry Group containing 19 convolutional layers, as described in [104].

Viola&Jones Refers to the algorithm of Viola & Jones for robust face detection using a boosted cascade of weak classifiers, as described in [119].

VOC Visual Object Class, used in the Pascal Visual Object Challenge.

VOT Visual Object Tracking Benchmark dataset.

YOLO You Only Look Once Detector. An algorithm for detecting object using only a single pass deep neural network, as described in [96].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Industrial automation is something we can no longer ignore. While automation of production processes has been something mostly used by large industrial enterprises in the past, nowadays even SMEs *(small and medium-sized enterprises)* are trying to benefit from the power of automation. There are of course several reasons for this. In latest years, technology for automation has heavily improved, providing easily accessible soft- and hardware at reasonable costs. Furthermore, automation is helping to improve the production processes, aiming at a higher quality standard, achieving more robust and uniform processes. It also allows companies to reduce personnel costs drastically and make the work environment safer. This leads to a large shift in job types, as seen in Figure 1.1. While in the past we focused more on manual labour at conveyor belts, we are now more and more focusing on technical personnel, needed to follow up on the automation systems and to act more as process managers.

Figure 1.1: Shift in job types, from manual to robotic apple sorting.

However, automation is far from fail-proof, and that is the main reason why we still need the human-in-the-loop mentality. While fully automated systems are very good at doing repetitive tasks with a very high precision, they still lack the ingenuity that humans have, especially when searching for outliers in the process. A human is capable of detecting small changes or differences with the greatest ease, while for automated systems, it is quite hard to detect these differences without a redesign of the system. This has led to a large research field, focussing on improving the existing human-in-the-loop systems *(which can be seen as semi-automated systems)* towards fully-automated production systems *(a human operator is in principle no longer needed).*

A nice example of this contrast between humans and automated systems can be found in inspection tasks. Given a set of objects from the same production line, a human is very good at detecting small cracks or faults, easily viewable with the human eye and this at incredible speeds. This even works even if the person has no prior knowledge of what an actual defect will look like. The human brain, in this case, is very strong at seeing these immediate differences. A fully automated system, however, is still lacking these self-learning capacities, requiring the need of teaching it how the defect will look like. This needs tailor-made algorithms by engineers, explicitly telling the system what to look for. And that is the hardest part, defining those exact properties of the defects in order to be able to automatically classify them.

In this dissertation, we want to touch upon a very small part of the complete automation process. The interdisciplinary field that focusses on gaining a high-level understanding of digital images and videos is called computer vision. Its main focus is to try and mimic the human visual system, and perform evenly well. This research field is needed to tell an automation system what is actually occurring in its surroundings. One of the main tasks of many automation processes is to automatically locate objects in an input image, captured by a camera system. This is frequently referred to as object detection, but in literature object classification and object categorization are also used. Object detection is thus the task of providing a user with the exact coordinates of a given object inside the camera coordinate system, which in turn can then be related to world coordinates by other techniques in the computer vision field.

A mistake frequently made in computer vision, both in literature as in industrial context, is the use of the concept of object recognition for object detection tasks and vice-versa. Therefore we think it is important to highlight this difference.

Object recognition is the process where we look for an exact instance of an object. We know beforehand exactly how the instance looks like and try to find this instance in a given scenery. Take for example a camera pointing at a coffee stand, where we want to detect coffee mugs. If we know that we are looking for a blue coffee mug with a black bird on top of it and an oval cup ear, then object recognition will locate that single coffee mug. However, it will ignore all other coffee mugs, even though they are part of the coffee mug class.

Object detection, also called object classification or categorization, aims to go one step further, and instead of focussing on a single object instance, rather look at a complete object class, with all the included variance. Different mugs are not visually the same *(different colours, different prints, different sizes, ...)*, but they all contain a cup and an ear to hold the cup. Object detection aims at generalizing the features describing the object class, and tries to find a detector that can detect every single coffee mug, without telling which mug specifically. This is exactly what our research focus will be, generating object detection approaches for general industrial relevant object classes.

In the remainder of this chapter, we give a concise summary of the main contributions of this PhD research. We also describe the outline of this dissertation, discussing briefly the contents of each chapter.

1.1 Problem statement

Grabbing academically developed object detection algorithms and simply applying them to industrial applications is not always feasible. There are several issues with the available object detection algorithms:

- Most algorithms are trained on publicly available datasets. These ensure that enough training data is available for specific classes *(depending on the focus of the dataset)*, in order to be able to learn a very stable and descriptive model for the object class. Collecting similar amounts of training data for completely new classes, is time-consuming and thus very expensive for any enterprise trying to get started with object detection.
- Academic datasets are in most cases completely irrelevant to industrial applications, as seen in Figure 1.2. E.g. the publicly available Pascal VOC dataset, contains object classes like a dog, a sofa, a chair, ... while industrial relevant cases are looking for object classes that directly relate to their application. This generates a mismatch between academic available data and models trained on that, compared to the actual desires of the industry.

- On top of collecting all that training data, one also needs to manually label them with coordinates and class labels of the actual objects in the given image. Again this is a very time-consuming and labour-intensive task. Academics benefit from already available datasets, and thus do not need to do this relabelling again.

- Training of more general 'in-the-wild' object detectors *(detectors that basically work everywhere)* can easily take days up to weeks. Commonly used training parameters are available in many academic publications, making it feasible to predict the actual training time. However, for a new industrial relevant case, there is no telling what parameters will work properly, not to mention it is impossible to estimate beforehand how long training the object detection model will take. This is not so feasible in a deadline-driven context.

- In more recent research, deep transfer learning, where an existing model is fine-tuned into a new object class detector, drew the attention of many researchers. However several of our experiments prove that this is only feasible when your object data is related *(e.g. a similar context or camera viewpoint)* to the dataset. If your data is utterly different *(e.g. dataset is frontal viewpoint from a car, but your data contains satellite images)*, then many transfer-learning techniques do not reach a stable object detection model and thus require the training of a model from scratch.

- Most techniques are developed in academic context, frequently using a Matlab environment. For industrial applications, this is a bad choice by far since Matlab requires a runtime license for each system on which the application runs. Combined with the fact that most open source academic implementations are not as well documented as it should be, it is very difficult for companies to simply implement these techniques on their own.

Figure 1.2: Examples of typical object detection classes in an academic context *(sofas, babies, bikes, aeroplanes, dogs, chairs, ...)* and industrial context *(orchids, pancakes, micro-organisms, bell peppers, ...)*.

All these issues with using off-the-shelf detectors convinced us that there is still a largely unexplored territory, with a focus on increasing the usability of these object detection algorithms for industrial applications.

We also noticed that even though the field of computer vision is drastically evolving and literally generating new algorithms each month, that a lot of application fields are lagging behind. Agriculture and medical fields are both great examples of places where automation is still being done with very basic and old computer vision techniques or even totally manually (e.g. microscope counting). These have only a very limited capability of generating satisfying results. Pixel colour based segmentation with manually set thresholds is still a very commonly used technique for object segmentation and object detection in many application fields, while the computer vision field moved on and now uses machine learning, and even more recently deep learning, to automatically learn the best separation between object and background.

Given the above insights, the general problem statement for this dissertation can be formulated as:

"Given the power of state-of-the-art computer vision based algorithms, we aim at providing easy-to-use object detection solutions for industrial applications, requiring a minimal amount of manual intervention, but ensuring very high accuracies and real-time performance."

1.2 Main contributions

Our research positions itself between academic research on state-of-the-art object detection algorithms and industrial ready-to-use object detection applications. We grab existing techniques from academics and adapt them to the actual needs of the industry. By doing so we close the valley of death, seen in Figure 1.3.

We notice that academically developed algorithms are in many cases not yet mature enough to get integrated into actual applications. Where academics invest a lot of resources in the fundamental research phase, once they have a proof-of-concept they tend to move on to a new research topic, and many times ignore the development phase. On the other hand, developers in an industrial context, prefer algorithms that are robust, stable and bug-free, so that they can pick it up instantly and move to the industrialization phase. They do not want to invest many resources until an algorithm is actually mature enough. To bridge that gap, and thus close the valley of death we focus on adding this extra layer of robustness into publicly available object detection frameworks.

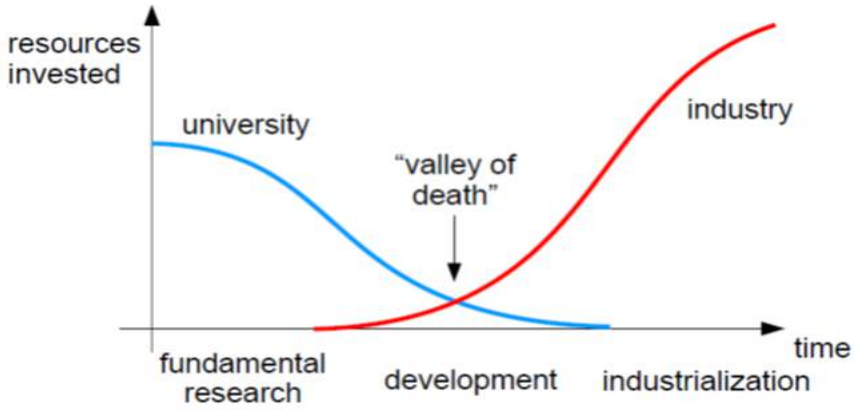


Figure 1.3: Illustration of the valley of death in application development.

The main contributions that we propose throughout the different chapters in this dissertation can be summarised as follows:

- We reckon that industrial applications are in fact way more constrained than their actual academic counterparts. While in academics we try to develop object detection algorithms that are as robust as possible given the variation in the object class *(e.g. pedestrians have a variety in clothes, physical build, skin colour, ...)* and application context *(e.g. background, lighting, ...)*, in industrial context, many of the possible scene and object variations are heavily controlled. This allows us to define scene and application-specific constraints that can help improve the performance of the object detection algorithms.

- Compared to academically developed algorithms, industrial software needs to be as real-time as possible *(hard constraint)*. We thus focus on increasing the processing speed of these algorithms whenever possible.

- Missing and wrong detections can have a huge impact in industrial set-ups. Therefore, we aim at building detection algorithms that are able to detect all objects in a given image or video sequence with a high certainty.

- In fully automated systems, the amount of manual labour should be minimized. Object detection techniques need a tremendous amount of manual labelling work. We tackle this issue by demonstrating an innovative active learning strategy. This allows us to only ask for manual annotations on a minimal set of images.

- Due to the latest trend in computer vision, being deep learning, and the tremendous drop in hardware costs, we investigate the possibility of using these very powerful algorithms for industrial object detection tasks. We prove that fine-tuning existing deep learning models to new detections tasks is a feasible solution for industrial object detection.
- We know that collecting enough valuable data for training object detection models can be a hazardous task. We investigate the relationship between accuracy and the number of actual training samples needed and prove that we only need to find a minimal set of valuable training samples to obtain highly accurate object detectors.

1.3 Outline of research

After this introductory chapter, where we outlined our research, discussed our actual problem to tackle and gave an overview of the main contributions, we discuss the related work in **chapter 2**. In this chapter we take a closer look at the evolution of object-based detection techniques in the past years. We discuss different approaches in object detection, motivate why we stick to specific frameworks and give an overview of the state-of-the-art in the field.

Chapter 3 discusses how we analysed industrial applications and discovered many possible scene- and application-specific constraints that can be used to improve object detection algorithms. We apply these scene constraints both as pre- and post-processing filter on top of existing object detection algorithms.

By integrating the knowledge of these scene constraints into the actual training data and training process, we mimic more advanced object detection techniques using the robust and well-known cascade classification pipeline in **chapter 4**.

Trying to solve the issue of the costly and labour-intensive manual annotation process, **chapter 5** discusses an active learning strategy we developed for our industrial object detection purpose. In this chapter, we aim to reduce the need for manual annotation as much as possible while maintaining similar accuracies.

Since deep learning is still a rising trend in computer vision, we investigate the possibilities of using pre-trained object detection models and model fine-tuning in **chapter 6**, to obtain even more robust and versatile object detection models.

In **chapter 8** we conclude our dissertation. We take a look at future improvements and possible expansions to this work, have a look at what new upcoming trends in computer vision could mean for our problem statement and what it will take to make them mature enough for industrial use. Finally, we give an overview of the industrial valorisation of this dissertation in **chapter 9**.

Chapter 2

Related work

In this chapter, we discuss in detail the related work to this dissertation. We first focus on the evolutions of object detection algorithms in section 2.1, where we highlight the current state-of-the-art in object detection and motivate why we selected specific algorithms as a baseline. We take a closer look at research that uses application-specific constraints to improve their results over unconstrained scenarios in section 2.2. In section 2.3 we discuss active learning and its use in application-driven object detection research. We conclude the chapter by looking at the selected frameworks in section 2.4 and discussing metrics to compare and evaluate object detection algorithms in section 2.5.

2.1 Object detection

Initial object detection solutions in computer vision applied a series of classical image processing filters on the input image to filter out image pixels that should belong to a specific object class. These segmentation based approaches require a user to set hard thresholds on a specific property of the image pixels, e.g. all pixels with an intensity value higher than 128 get a true label, all others get a false label. Each threshold set on a specific property, results in a binary mask of possible candidate pixels. By then combining these masks, possibly adding a weight to each mask to represent the relevance of the property, people obtained a final mask of object pixels. These are then automatically grouped by applying connected component analysis and using erosion and dilation filters to remove noisy pixels. However complex object detection cases in unconstrained environments made it quite difficult to find these specific properties that allow

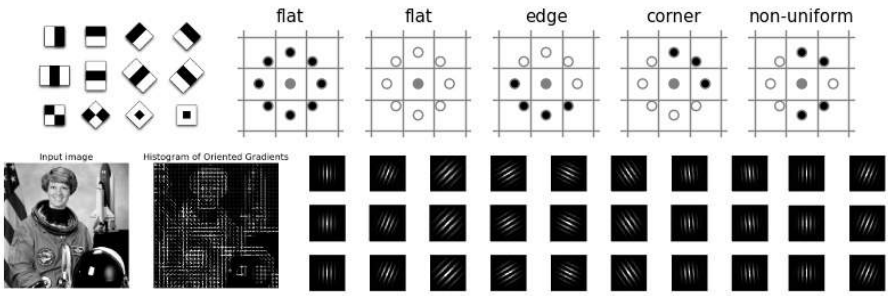


Figure 2.1: Examples of variation invariant pixel representations: *(top left)* Haar-like wavelets *(top right)* Local Binary Patterns *(bottom left)* Histogram of Oriented Gradients *(bottom right)* Gabor filters.

separating object pixels from background pixels, leading to a time-consuming and expensive quest of finding the problem defining properties.

Nevertheless, several companies we came into contact with at the beginning of this PhD research were still actively using this segmentation- and threshold-based approach. To improve these manually thresholded segmentation based approaches, people started looking at the power of machine learning. These supervised techniques use labelled data of the object class and the non-object class to learn how to set the optimal threshold, removing the need to set a manually selected threshold, over a given validation dataset. However, in this case, people still need to look for the significant pixel properties that allow for learning this optimal threshold.

The biggest drawback of threshold-based techniques is the influence of variation. Too much variation in for example lighting, shape or colour, will result in a failing object segmentation approach and will either allow some of the background pixels to be classified as object or vice-versa. To cope with these variations, research was pushed in a new direction, using variation invariant features, automatically learned by machine learning algorithms. Some of these variation invariant pixel representations are *Haar-like wavelets* [119, 65], *Local Binary Patterns* (LBP) [43, 4], *Histogram of Oriented Gradients* (HOG) [16], *Gabor filters* [48], ... as illustrated in Figure 2.1.

The general approach of automated learning systems for object detection is quite straightforward. It requires the generation of a set of training data, both object images and background images. For the object images, one needs to manually provide an annotation for each object in the image. This is done by drawing a bounding rectangle around the object and storing its coordinates

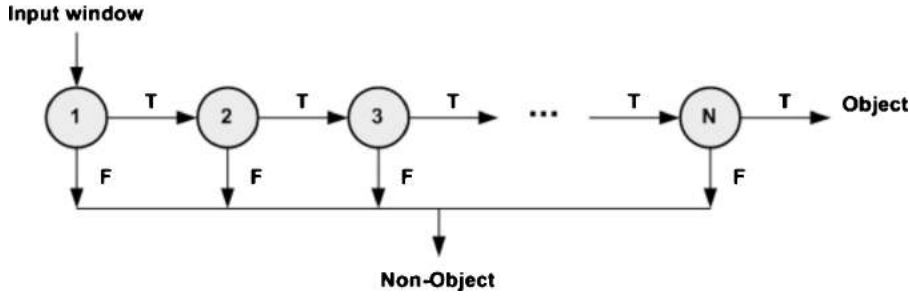


Figure 2.2: Viola and Jones algorithm: early rejection principle.

relative to the image dimensions. For each annotation, a feature representation is generated and passed to the classification algorithm. For the background data, random samples at model size are grabbed from images not containing the object. Once all data, both object and background samples, is transformed to the correct feature representation, a classification algorithm is used to learn the best separation between both classes in the high dimensional feature space. To ensure that these techniques do not only work on a fixed scale, being the model dimensions, images are parsed into an image pyramid, where each image is up-scaled and down-scaled. Once a detection with a fixed model size is found on a scaled part of the pyramid, the found detection is warped back to the original image size, allowing for multi-scale object detection into a single model.

A first novel groundbreaking approach in object detection is the boosted cascade of weak classifiers, presented by Viola and Jones [119] in 2001. Originally developed for robust face detection, it was quickly adapted to work for general object detection [65] in 2002. They use Haar-like wavelets as feature representation, which is basically a subtraction of pixel intensities of neighbouring pixels. This type of feature can be calculated quite fast by using integral images. On top of this feature representation, an adaptive boosting algorithm *(AdaBoost)* [35] is used to train a cascade of weak classifiers. By applying a combined sliding-window and image pyramid approach, an image is efficiently processed in every location and on different scales. The power of the cascade structure is that many windows are only evaluated by a small set of weak classifiers, able to reject a lot of negative windows, and thus increasing processing speed drastically. This is called the early rejection principle, as illustrated in Figure 2.2.

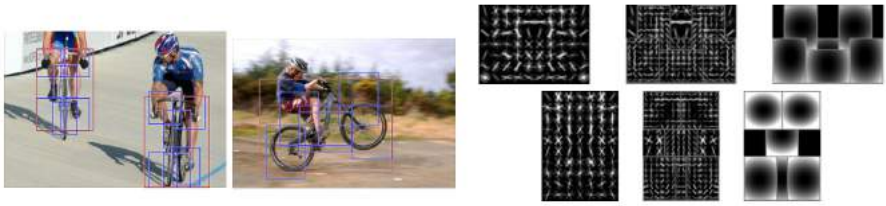


Figure 2.3: Deformable Parts Model for bikes as suggested by Felzenszwalb et al. [31]. *(left)* Frontal and side model applied to images *(right)* Root model + part models + distributions of allowed deformation.

The approach proposed by Dalal and Triggs [16] in 2005 was the next step in object detection and more specific in pedestrian detection. By using *Histograms of Oriented Gradients (looking at local image gradients)* in combination with a *Support Vector Machine* (SVM) classifier [15], they automatically learn a detector that is able to separate pedestrians from the background.

Until then detectors always assume, given a set of features to represent the object image, that the features are in a fixed position in relation to each other. This only allows modelling rigid, non-deformable models for object detection. Given objects like pedestrians, one could argue that those rigid models are actually under-performing since a pedestrian is a highly deformable model. Therefore the research of Felzenswalb et al. [31] introduced in 2008 the concept of *Deformable Parts Models* (DPM) for object detection. They make a combination of a root model, a rigid model of the complete object, but extend that with parts and a deformation relation between those parts. Each part is a rigid model on itself, and can be seen as logical object parts like a leg, a head, a torso, ... This introduction gave object detection again a large performance boost. Figure 2.3 illustrates this. On the left-hand side, we see both a frontal and side model view for detecting bicyclists. On the right-hand side, we can see the model separation for both views, starting with the rigid root model, then the rigid part models and finally the allowed deformations on the parts.

One can argue that only using a single invariant feature representation will never reach the full potential for object detection. That being said, we saw an explosion in techniques combining multiple invariant feature representations in a single model, improving object detection accuracies. However, brick walls were still hit. To reach an even better performance, Dollár et al.. suggested in 2009 [26, 25] to take a step back, and acknowledge that for example colour information, although highly influenced by lighting conditions, still has a major performance boost when added to these invariant feature representations.

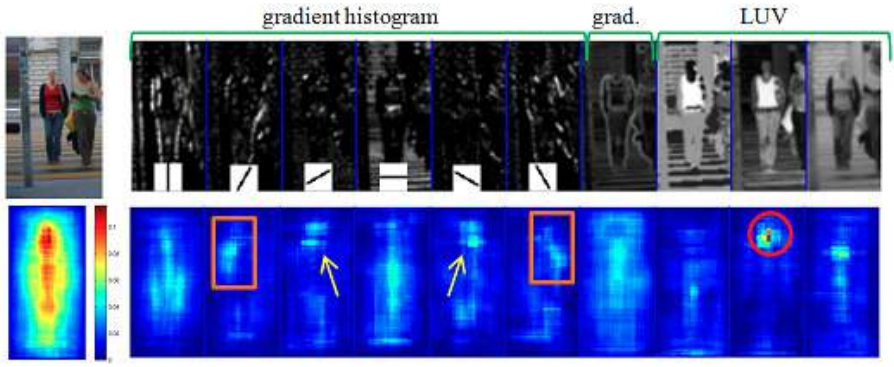


Figure 2.4: Integral Channel Features: adding LUV channels triggers high response on face area as shown by Dollár et al. [26] over a large training set.

For example, the face area has a distinctive feature in the LUV colour space, as illustrated in Figure 2.4. They introduced a technique called *Integral Channel Features* (ICF), where a large set of feature representations is combined. The underlying cascade boosting system then smartly selects the most discriminative features from different channels, instead of using all of them, to build a very robust object detection algorithm.

On top of this work, a lot of variants were proposed by Benenson et al. in 2012 and 2013 by either replacing the combination of feature channels, changing the way features are selected from different feature channels [6, 7], or replacing the classifier back-end to something like e.g. random forests.

The largest performance boost arrived with the upcoming trend of deep learning and the drop in the cost of *General Purpose Graphical Processing Units* (GPGPU). The introduction of deep *Convolutional Neural Networks* (CNN) in 2012 [55] and its many variants for object classification and detection pushed the performance limits of object detectors through the roof. The ability to learn feature representation from raw pixel input data by applying convolutional filters removed the manual handcrafting of discriminative feature representations. Furthermore, these learned CNN features proved to be more effective than any handcrafted feature out there. Since a complete CNN pipeline (see Figure 2.5(top)) was still very computationally demanding, initial research in object detection focussed on combining these CNN features with classic classifiers like SVM [38], *Boosting* [122], DPM [101] or *Random Forests* (RF) [41], as shown in Figure 2.5(bottom).

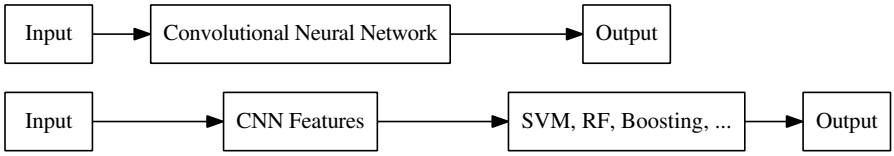


Figure 2.5: Different options of convolutional pipelines.

However, in recent years, object detection using deep learning *(also called deep object detection in the remainder of this dissertation)* took a huge step forward. Due to the rise in computational power with affordable GPGPUs, one is capable of using a complete convolutional pipeline for object detection, so both feature representation and classification. Several state-of-the-art approaches are based on this principle.

However, one issue still remains. Since *Convolutional Neural Networks* process image patches, doing a multi-scale and sliding window based analysis takes a lot of time, especially if image dimensions increase. Combine this with the fact that deeper networks achieve a better detection accuracy, very deep networks like *VGG19* [104] and *InceptionV3* [110] can easily take several seconds for a VGA resolution image. To increase the execution speed of these algorithms, researchers introduced the concept of region proposal techniques. Basically, these techniques are fast and lightweight filters that pre-process images, looking for regions that might have promising content. Instead of supplying the convolutional chain with multiple millions of windows, the region proposal algorithms reduce this to only several thousands of windows.

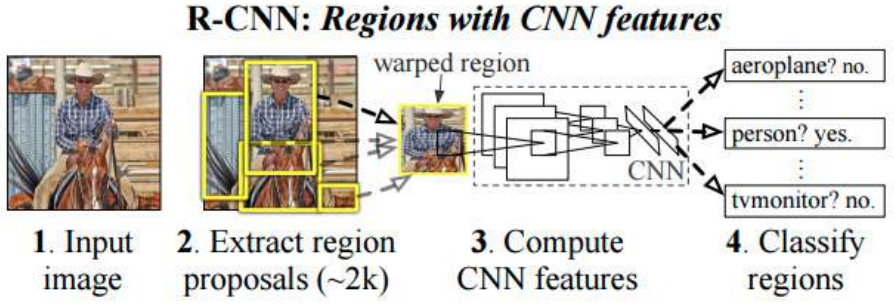


Figure 2.6: Region proposal networks to obtain faster convolutional neural networks for object detection as proposed by Ren et al. [98].

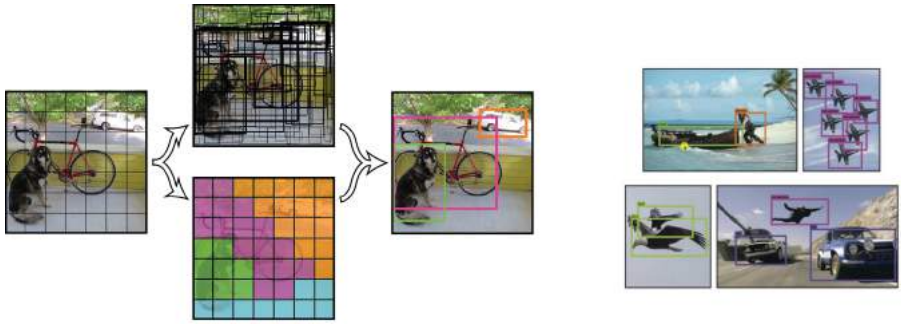


Figure 2.7: Single pipeline deep object detectors: *(left)* 'You Only Look Once' - detector (YOLOv2) [96] *(right)* 'Single Shot Multibox' - detector (SSD) [67].

Region Proposing Convolutional Neural Networks (R-CNN) [98] use a shallow region proposal network, as illustrated in Figure 2.6, on the GPU that uses mutual information of the actual detection pipeline. By doing so we allow proposing a limited set of region candidates towards the classification part of the deep learning pipeline. This enabled to run very deep models like VGG19 at 10FPS, giving a 30x speed improvement towards the original implementation.

The *'Single Shot Multibox'*-detector (SSD) [67] introduces an algorithm for detecting objects in an image using only a single deep neural network, immediately grouping the convolutional output activations and returning object boxes. The *'You Only Look Once'*-detector (YOLO) [96] implements a similar approach, using anchor points that allow learned aspect ratios around pixel areas that have a high response. Both detectors remove the need for extra region proposal networks and perform bounding box prediction and class probability generation in a single run through the network, as illustrated in Figure 2.7.

Even as we speak, the field of object detection keeps growing and producing new and more effective detection models, e.g. the work of Redmond et al. [97] delivered a 9000-class real-time object detector. It seems that deep learning has, for the time being, conquered the top spot and is there to stay for now.

2.2 Application-specific constraints

Using constraints in a computer vision application to obtain better end results is not something completely new. Especially in machine vision, where constraining the application by introducing a controlled lighting source, a pre-known trajectory of the object, a pre-known 2D shape, ... so that basic threshold

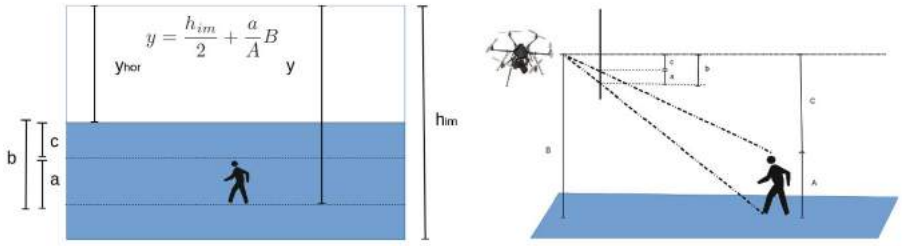


Figure 2.8: Ground plain assumption illustrated as shown by De Smedt [19].

based segmentation will work, is a common habit. If there is a lot of application-specific variation then applying these more constrained techniques like the segmentation approach suggested by Lee et al. [59] will be nearly impossible.

We highlighted the advantage of using application-specific scene constraints *(e.g. a constant illumination, a fixed camera position or a known background)* to improve state-of-the-art object detection algorithms in our first publication [88]. The paper suggests using the knowledge of the application-specific scene- and object-conditions as constraints to improve the detection rate, to remove false positive detections and to drastically reduce the number of manual annotations needed for the training of an effective object model. Some examples from the literature are given below.

The work of Mathias et al. [71] clearly shows that the combination of object categorization and known object constraints works for robust traffic sign classification. In this case, the lighting conditions are quite variable, but the colour and shape properties of the object itself are quite stable. Furthermore, research like the work of Cho et al., Peng et al. and Dibra et al. [14, 84, 22] describes the use of a ground plane assumption for 3D modelling and multiple camera view processing. This ground plane assumption assumes that pedestrians in a relative position to a fixed camera (for example mounted on the top of a car) must have a fixed average size. The same assumption is for example used in the work of De Smedt and Hulens [19] where they apply a similar ground constraint for efficient pedestrian detection from UAVs, given a known height at which the drone is flying, as shown in Figure 2.8.

In every industry-relevant application we developed, we firstly investigated the state-of-the-art on similar techniques. Many times this resulted in research pointing at the importance of using scene constraints, which we will further investigate in chapter 3, or heavily over-relying on actual constraints. We will discuss this in detail in a separate related work subsection in each of the following chapters and sections.

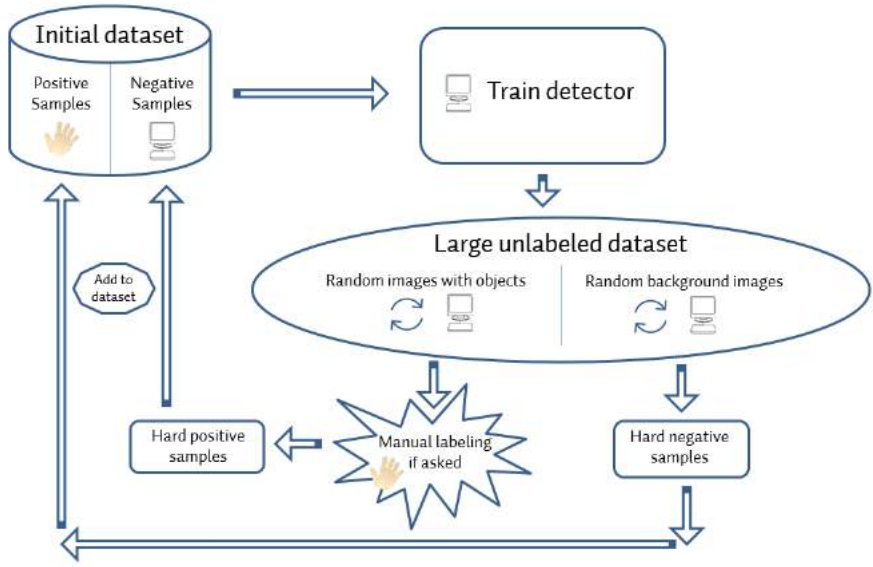


Figure 2.9: The active learning principle illustrated for object detection, focussing on selecting the most valuable training samples.

2.3 Active learning

Supplying large amounts of training data to machine learning algorithms allows learning very complex object detection models. However, the downside is that both in gathering positive and negative training data, it is very difficult to tell which new sample will actually improve the efficiency of the detection model. In order to cope with this problem most academic developed techniques just gather a tremendous amount of training data, ensuring that most frequently occurring situations are captured by the dataset. This works when your target object is an object class that has active research and thus finding training data is simply a task of combining publicly available datasets. However, if your application field is completely unrelated to the currently investigated academic application fields, then you are left in collecting all that data yourself. And since the saying *'Time is money.'* still stands in industry, putting employees on this time-consuming and resource-expensive task is in many cases the reason why the industry does not use machine learning algorithms.

If we want to drastically reduce the amount of manual interventions, needed for learning a machine learning based object detection model, then we should focus on getting the most valuable samples labelled first. In order to decide which samples are actually valuable to be added to the process, we will apply a technique called active learning in chapter 5. The idea of active learning is to use an initial weakly trained detection model, based on a limited set of training samples and use that model to predict for us which samples are actually valuable *(close to the decision boundary)* for adding in a second training iteration and which are not *(no ambiguity in labelling)*. This principle can be seen in Figure 2.9. We make a distinction between hard negatives and hard positives as explained below. Furthermore, the advantage of active learning is that it limits the amount of manual labour since we only need to provide labels to new training samples that add extra knowledge to the trained classifier.

Applying active learning in object detection has been done before. The work of Kapoor et al. [52] combines active learning with Gaussian processes in order to build an effective object categorization pipeline. While using a very small dataset for a single category, they are still able to obtain a significant boost in classification performance compared to using a predefined large set of training samples, even indicating that using too many general samples, will make the model less discriminative than only using qualitative samples.

The work of Lia et al. [61] on adaptive active learning tries to tackle a common issue when using active learning strategies. The query selection strategy is key to create a robust active learning algorithm. With a large amount of unlabeled instances, these techniques can become prone to querying outliers. Their adaptive active learning strategy uses a selection mechanism that combines an information density measure and an uncertainty measure to select critical instances to label for image classification tasks, making it very suitable for object and scene recognition tasks.

In chapter 5 we use this active learning strategy to drastically reduce the number of annotations needed when collecting valuable training data. We use these concepts to train initial weak classifiers with a very limited set of training data, use those models to iteratively search for valuable training samples *(both object and background samples)* and only ask for manual input *(through label correction)* if the classifier itself is not sure of its decision to label it as object or background sample.

2.4 Selected frameworks

A final section in this related work chapter discusses how we selected the framework as a base of most of our developed research. When starting our PhD research in 2013, several approaches were available, but most of them were heavily optimized for either face or pedestrian detection tasks. Combined with the fact that many of these algorithms were only available as proof-of-concept software in Matlab package form, this limited our possibilities when developing industrial relevant object detection applications.

The OpenCV framework [12], standing for open-source computer vision and now owned by Intel, was a welcome alternative to what was around. It included several implementations of robust machine learning algorithms like boosting, support vector machines, random forests, ... that could work as efficient classifiers for our industrial relevant datasets, based on the on that moment state-of-the-art object detection algorithms available. Combined with good documentation, an intuitive C++ interface and an active community, the decision was made to continue our research on the base of this package.

After some initial test set-ups to compare different object model detection interfaces in OpenCV, we decided that the robust and proven implementation of the Viola and Jones algorithm [119] was the best candidate to continue our research on top of. It allowed for applying scene constraints as pre- and post-filtering, as discussed in chapter 3, as well as integrating knowledge in the actual training pipeline (discussed in chapter 4) mimicking the behaviour of the more powerful ICF technique [26]. Furthermore, it allowed us to focus on implementing extra functionality, rather than building an efficient boosting system again from scratch. Given the fact that the software was also released under a BSD license, this was the perfect candidate for industrial applications, since licensing is no longer an issue.

However, we did not simply stick to adaptive boosting and building robust cascades of weak classifiers for our industrial object detection tasks. Once our research group implemented a standalone C++ version of the Dollár *Integral Channel Features* approach [18] we also decided to compare this against our boosted cascade models, as seen in chapter 4 section 4.2.

Finally, by the end of my PhD research, we could no longer ignore the exponential growth in deep learning algorithms for object detection. We considered available deep learning frameworks and found an easy to adapt framework based on C and CUDA, called Darknet [95], which can easily be integrated with our existing software. Based on the YOLOv2 object detection model [96] we applied several deep learning object detectors to industrial cases, proving their use, even when having only access to very limited datasets, as discussed in chapter 6.

2.5 Comparing different detection algorithms

Since our dissertation handles multiple object detection algorithms and several trained models, we need to establish a baseline approach for evaluating all our trained models.

2.5.1 Multi-scale sliding window and non-maxima suppression

Before digging into evaluation metrics, we first need to elaborate on how object detections algorithms return their detections. Object detectors, in general, are trained on image patches either containing the object *(positive training samples)* or not containing the object *(negative training patches, also called background patches)*. However, when using an object detector in an application, we desire to supply full images to the system, and not precut image patches.

In order to allow our detection algorithm to apply localization of patches classified as an object, we apply a multi-scale sliding window based approach. This means we move a patch, with the same size as the training patches, over the image, with a pre-set step size in both x- and y-direction. Each patch is subsequently evaluated by our model and given a detection probability *(a certainty indicating a patch belonging to the object class)*.

However, this sliding window only provides us with localisation properties. Using a fixed size window only allows detecting objects at a fixed scale. In applications, however, objects can appear in different scales and we would want to avoid training a specific object detection model for each scale that can occur. For this, we up-scale and down-scale the image by a predefined scaling factor. Once a detection is found in a resized image, the detection bounding box is transformed back to the original image using the scaling factor specific to that resized image.

Of course, performing a multi-scale sliding window based approach yields multiple slightly shifted detections around an object. To solve this problem we apply non-maxima-suppression (NMS) on the returned detections. Based on an overlap criteria, bounding boxes are merged and only the highest score of the merged bounding boxes is kept and assigned to the final resulting bounding box, as seen in Figure 2.10. When deciding on the overlap criteria, one has to keep in mind that objects close together and slightly overlapping should still be detected as single objects, instead of returning a single melted-together detection. A detailed study of NMS, selecting the correct overlap threshold and its influence on detection accuracy can be found in the addendum to the work of Dollár et al. [27].

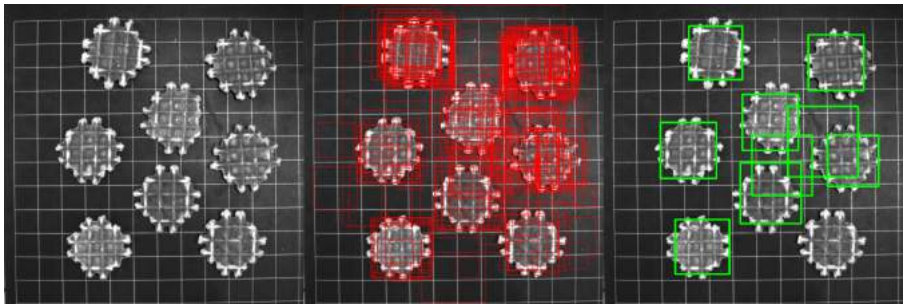


Figure 2.10: Example of non-maxima-suppression: *(left)* original *(middle)* after detector *(right)* after non-maxima suppression.

2.5.2 Precision-recall curves

Comparing the accuracy of different object detection algorithms is frequently done in the literature using precision-recall curves (PR-curve), and is done as follows. On a frame-by-frame basis, the algorithm evaluates each detection that is returned by the object detection model. For every detection that is matched with a manually labelled ground-truth annotation, we count a true-positive detection (TP). A detection that is not matched by a ground-truth annotation, triggers a false-positive detection (FP). If an object that has a manually labelled ground-truth, is not detected by the model, we count a false-negative (FN). We consider a matching ground-truth annotation and a detection, again based on an overlap criteria, called the intersection-over-union (IoU). Whenever the IoU has a value which is larger than 50%, two bounding boxes are considered a match. However, in application-specific scenarios, this value can be lowered or increased as needed.

Given a ground-truth bounding box (A) and a detection bounding box (D), the IoU is formulated as:

$$IoU = \frac{area(A \cap D)}{area(A \cup D)} = \frac{\text{intersection}}{\text{union}} \tag{2.1}$$

Based on the found matches and the returned values for TP, FP and FN, we can now formulate precision (P) and recall (R) as follows:

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

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

The precision indicates the percentage of correct detections *(the object the model is trained for is detected)* versus all retrieved detections. At the same time, the recall indicates the percentage of objects in the complete dataset that are actually detected.

Each detection algorithm couples its detections with a probability, a value identifying how certain we are that the retrieved detection actually belongs to the object class we are looking for. By varying a threshold over this probability we can remove detections that do not agree to the threshold and thus change the number of TP, FP and FN. Iteratively changing the threshold, generating specific precision-recall positions, leads to a precision-recall curve.

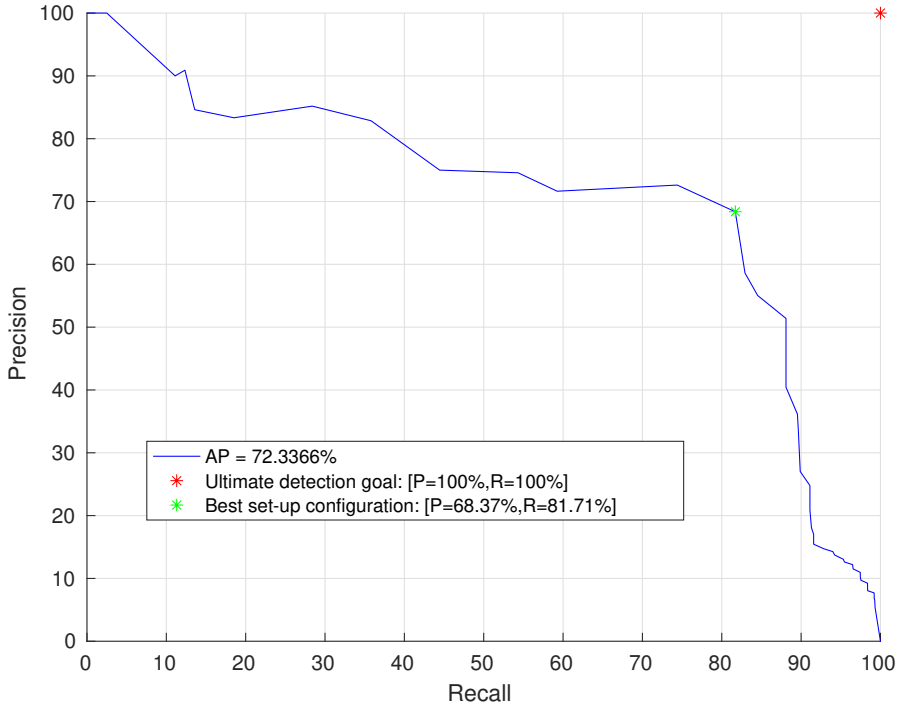


Figure 2.11: Example of the precision-recall curve with average precision (AP), the optimal solution (red star) and the best configuration (green star).

If the threshold is set low *(sloppy)*, more objects will be detected at the cost of an increase in false detections *(a higher recall at a lower precision)*. If the threshold is set high *(strict)*, fewer objects will be detected while the number of false detections also decreases *(a lower recall at a higher precision)*.

An example of a resulting PR-curve can be seen below in Figure 2.11. The red star indicates the best obtainable detector with a precision and recall both at 100%. The green star indicates the best set-up specific solution if we consider the closest distance to the top left corner. However, we proved with several applications that the optimal position can differ, depending on what your application prefers *(obtaining a higher precision or a higher recall)*.

2.5.3 Evaluation metric for deeply learned classification networks

In chapter 6 we introduce the concept of using a classification network in combination with a manually constructed sliding window based approach, to compare against actual object detection networks. In literature, where classification networks are validated over large public datasets, the metric of topX-accuracy is being used, instead of the more classical precision-recall curves, where X can be replaced by a number of classes. For large multi-class classification challenges, top1-accuracy and top5-accuracy are the most frequently used metrics. Since the applications in chapter 6 only cover two-class classification models, we only use the top1-accuracy as an evaluation metric, described in the equation below.

$$\text{top1-accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

The top1-accuracy is the ratio of the correct detections *(true positives and true negatives)* over the total number of patches provided to the classification system.

In the following chapter we start by discussing how we used scene- and application-specific constraints to our boosted cascade of weak classifiers pipeline in order to obtain efficient, industrially relevant and real-time object detection algorithms, supported by actual industrial cases.

Chapter 3

Introducing object- and application-specific scene constraints

The work presented in this chapter was published at the MVA 2015 conference [89], the EMBC 2015 conference [85] and the VISAPP 2016 conference [90].

General academic research on robust object detection starts from one of the frequently used cases, e.g. pedestrians or faces. This is quite understandable since these classes already contain a lot of challenges one will need to tackle in any other object detection task, like a variety of poses, colour combinations, illumination differences, ... This results in academic focusing on detecting objects in-the-wild, which means that an object should possibly occur in any possible condition with any possible kind of background information. Given the fact that trying to integrate all this variation leaves most state-of-the-art object detection algorithms with an accuracy of about 95%, we are sure there is still room for improvement.

However, many industrial applications specifically decide to partially constrain these in-the-wild conditions, in order to obtain higher accuracy rates *(aiming for accuracies of 99.9% and higher)*, that are more suitable for industrial applications. With an accuracy of 95%, we are still unable to detect five out of a hundred objects, which is unacceptable in industry, taking into account that most automated production processes work with thousands up to millions of object candidates a day.

These introduced application specific scene constraints, ranging from fixed and well-defined illumination sources to a known background *(e.g. a conveyor belt)*, can be combined with existing state-of-the-art object detection algorithms as either pre- or post-filtering steps to push the accuracy over current limits. One could argue that once you know for example the background, that object detection algorithms are overkill, and segmentation based approaches could suffice. Take for example the case of detecting chocolate figures on a conveyor belt. This belt is cleaned once a day, so at the beginning of a shift, it is clean and blue, making segmentation a valid candidate. However, after a couple of hours of running, the belt gets cluttered with leftover chocolate parts, and thus the algorithm starts failing. Stopping the belt multiple times a day is economically not manageable. One can see that in these cases a robust detection algorithm, capable of coping with this cluttered background, will do the trick.

The following subsections will discuss in detail three cases where we developed scene-specific pre- or post-filtering techniques to push the initial object detection accuracies one step further. Section 3.1 discusses the visual detection and species classification of orchid flowers in the context of an automated sorting installation. This is followed by a study of using object detection algorithms in elderly home care, in section 3.2. Here we used automated techniques to define if an elderly is using a walking aid or not, in order to obtain a better estimation of their walking capabilities, directly related to their health status. Section 3.3 discusses the topic of automated detection and privacy safeguarding of pedestrians in mobile mapping images. Finally, section 3.4 will conclude on how to use scene constraints as pre- and post-filtering for improving object detection accuracies.

3.1 Visual detection and species classification of orchid flowers

With more than 100.000 different Phalaenopsis orchid flower cultivars, we notice a large intra-class variation in shape, size, colour and pattern. Object detection techniques can build a single model for flower detection based on this heavy varying object data. Our application exists of an industrial orchid grading machine, in which the orchid plants are passing by a set of cameras. The orchid itself is rotated 360 degrees to capture the plant from all possible sides, as seen in Figure 3.1. In each view of the orchid plant, we want to detect all frontal facing orchid flowers. With only a limited set of annotated training data at hand *(only 250 orchid flower samples)*, we investigate the possibility of achieving high detection accuracies on a larger validation set.

Figure 3.1: Example of the input data for the given orchid flower detection and classification pipeline, presenting a 360-degree view of the complete orchid.

After collecting all individual orchid flowers, the flowers are passed to a classification pipeline, which determines which orchid cultivar we are dealing with. Again this classification pipeline is trained on a very limited set of data *(5 classes with on average 15 training samples per class)*.

The application itself has several set-up-specific constraints that make the training of a detector and classifier easier. First of all, there is a controlled lighting, which restricts the intra-class appearance variation, and thus the number of negative examples. Furthermore, it also allows using a colour specific image description for each species. Since colour descriptions will not be unique for all those orchid species, we will combine the colour description with an appearance based classification of the flower, based on the visual characteristics like texture, edge content, dominant colours, etc.

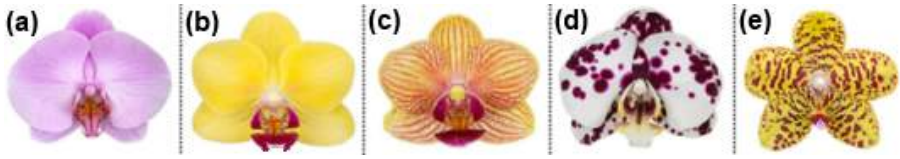


Figure 3.2: Example images of all five orchid flower classes. (a) Uniformly Coloured (b) Coloured Lip (c) Striped Pattern (d) Spotted Pattern (e) Speckled Pattern

Secondly, the position of the camera is known, and thus the scale range of the orchid flowers is known, which leads to an effective reduction of the huge object search space based on the image pyramid [88]. Finally, we have a known blue background, as clearly noticeable in Figure 3.1, which ensures there is no background clutter.

For classification, we propose to look at a specific set of visual features like colour differences, dotted patterns, radial lines, etc. After the flower detection phase, we efficiently combine all this information into a decision tree of binary support vector machines that succeeds in separating all provided orchid flowers into five texture based classes as seen in Figure 3.2. Combined with a subsequent unique colour description this class label successfully separates all given cultivars.

The remainder of this section is organized as follows. Subsection 3.1.1 discusses related work on orchid detection and classification, while subsection 3.1.2 explains in detail how we detect orchid flowers by using a scene constrained boosted cascade of weak classifiers. Subsection 3.1.3 elaborates on our flower species classification approach using binary support vector machines. Finally, subsection 3.1.4 wraps up some conclusions.

3.1.1 Related work on orchid detection and classification

In the application of orchid flower detection, the variety of species is huge *(shape, pattern and colour variation)*, which makes it impossible to use segmentation based approaches like suggested by Nilsback and Zisserman in [77]. In this work on the multi-class classification of flower species, they introduce a smart flower segmentation on shape and colour. However, they do not have a controlled lighting condition and do not discriminate different cultivars of the same flower species, like in our application.

Object categorization techniques might pose a solution to the problem. They have been used before for robust object detection in situations with a lot of scene variation *(lighting, occlusion, clutter, ...)*. In this specific case, the situation is somewhat different, due to the huge amount of object variation *(there are 60 Phalaenopsis species with in total more than 100.000 different cultivars)* but with a uniform, controlled lighting source, a known camera position and a known background colour. These application specific parameters can hopefully limit the amount of training data needed to obtain a robust orchid flower detector.

More recently Sfar et al. [102] describe an approach for efficiently classifying scanned leafs and orchid flowers, using an approach based on vantage feature frames. Their research is focused on a different variety of orchid flowers and due to the scanned image, the shape of the flower is far more unique than in our

application. The work also suggests creating a large set of boundary conditions in order to ensure a successful classification and a high recognition rate. This is exactly what we try to avoid since these boundary conditions cannot be achieved in our case of industrial flower classification.

3.1.2 Orchid flower detection

The first step in processing orchid flowers is detecting where the actual flowers are in the given input image. Without successful detections, we cannot pass the detected flowers to the classification pipeline. Due to a limited set of training samples, and moreover a very limited set of positive training samples compared to a large set of negative samples, it is important to select an object detection technique fit for this specific task.

Applying the HOG+SVM approach suggested by Dalal and Triggs [16] proved infeasible due to the very limited positive training sample dataset available. The trained detector does not seem to be able to learn an accurate separation between object and background patches. The detector output generates unsatisfying results and there is not enough data to generalize a proper model. Also, the HOG+SVM approach heavily depends on a balanced set of positive and negative training samples to increase its performance, which is not the case for the data we have at hand. The cascade classifier approach suggested by Viola and Jones [119] shows promising results given a very limited dataset, being able to generalize a model, even if a limited positive sample training set is available. In addition to that, the framework uses features, being Haar-like wavelets or Local Binary Patterns [43], that in our specific application seem to generalize better over the given dataset, compared to for example HOG features.

We take the basics of the Viola and Jones framework and adapt it to our needs of generic object detection, and more specifically, in this case, the detection of orchid flowers. We combine the open source training and detection interface provided by OpenCV [12] and make a complete training and detection framework for any given object class, based on the experiences collected in this application.

Training the object model

In order to build a robust orchid flower model, we collect a set of training images. As positive object images, 252 orchid plant images are grabbed from the industrial pipeline which already has a fixed camera set-up inspecting the plants. From these images, each flower is manually annotated with a bounding box, extracted and resized to an average size of 48×56 pixels. The original images, with the annotated pixels blacked out, are used as large background

images. From those images, a set of controlled samples, with same dimensions as the positive training samples, are grabbed as negative training examples.

On each training positive and negative training sample, the LBP or Haar-like wavelet features are calculated. Over this large set of collected features, a weak classifier per feature is created as a single depth binary decision tree. The AdaBoost algorithm then decides which trees are the most discriminative *(resulting in the best separation between positive and negative samples)* on the given training data. The final model is built by using 250 positive image samples of several orchid flower classes for each stage of weak classifiers and 2000 randomly sampled negative image samples. The model contains a set of 57 weak classifiers. Weak classifiers are combined in stages until each stage reaches a maximum false alarm rate of 50% while guaranteeing a minimum hit rate of 95% on the positive training samples before moving on to the following stage. The training takes around 20 hours on a dual-core processor with 8GB RAM.

Due to the known illumination and camera position, an application specific detection model for our industrial case is created by the training phase. This model will not work in any other set-ups used for orchid detection. It is highly dependent on the given background set-up and the way the orchids are presented to the system in relation to the camera set-up.

Object detection using the trained model

The general approach when using a cascade of weak classifiers is to run through the whole image pyramid, a step-by-step downscaled version of the original input image, with a sliding window equalling the models' dimensions. This allows us to perform a multi-scale object detection using only a single-scale object model. Unfortunately, the search space for object candidates turns out to be enormous when the size of the input image grows. Based on the orchid flower model that was trained and the knowledge of the camera set-up in this industrial application, we applied several restrictions to the detection pipeline, in order to efficiently reduce the search space. We start by increasing the image pyramid scale step and add restrictions to the object candidate size by limiting the scale range.

On top of reducing the search space and thus decreasing processing time for a single image, the reduction of the candidate space also helps to drastically reduce the number of false positive detections. A constant and diffuse lighting leads to an effective foreground-background segmentation, further reducing the search space of object candidates, resulting in a fragmented image pyramid, as we discussed in our earlier work [88].

A final improvement we applied is related to the application specific requirements and can be achieved by applying a higher detection certainty threshold on the

Figure 3.3: Orchid flower detections without false positives.

.

final stage of the boosted cascade of weak classifiers. Since we only aim to correctly classify a complete orchid plant, it does not matter if we miss some flowers with our detection in a single view of the plant. By combining multiple views of a single orchid plant for processing, which is automatically supplied by the orchid sorting set-up, we are able to increase the final stage threshold, so that only flower detections with a high certainty get accepted and the number of false positive detections *(a detection in the image triggered by pixels not containing an actual orchid flower)* is reduced to zero.

Detection results

The fact that we obtain a well-performing object detector with a very small training set, by carefully selecting the correct training and detection parameters, is quite impressive. On the complete validation dataset of 360 test images, each containing several orchid flowers, not a single false positive detection was reported, while still maintaining at least a single detection for each orchid plant image. An example of these detection results without false positive detections can be seen in Figure 3.3.

3.1.3 Orchid type classification

Only detecting orchid flowers in a given input image does not yet solve the problem of identifying the exact Phalaenopsis flower cultivars. We present a complete pipeline for classifying flower cultivars towards a predefined class using a binary support vector machine set-up and majority voting system. Since there are so many cultivars, it is impossible to learn a single classifier that is able to classify each and every one of them. Based on the visual characteristics of the flowers, we divide the cultivars into five pattern based orchid classes, as

Figure 3.4: Example of orchid flower detections being segmented out for orchid type classification.

already shown in Figure 3.2. Combining this with a colour based descriptor of the flower itself, allows us to decide which cultivar is presented to the system.

Visual characteristics and features

We start by segmenting out the flower from the background, based on the images retrieved from the orchid flower detector. Our application has the advantage of a known set-up, including a clear blue background, which makes the segmentation task a lot easier. On top of that, we filter out dark green and dark brown regions which could be parts of the plants, like stokes or flower buds, partially occluding the actual flower. One could argue that this is only valid when evaluating on a dataset that does not contain dark brown- and green-like coloured orchid flowers, which would otherwise be segmented out. However, the final industrial installation also captures a fluorescence image under UV lighting, which yields a high response to flower buds, leaves and stokes, but not on flowers. This could be used to replace the green and brown based colour filtering. All this leads to a segmented flower region image, which can then be passed on to the classification process. This is visualised in Figure 3.4. Histogram equalization is applied to the RGB colour channels to improve contrast.

Considering the five orchid flower classes mentioned in Figure 3.2, we generate a set of visual observations about the flower appearance. As a general characteristic for all orchid classes, when a flower has multiple colour ranges, then there are at most two colours, called a foreground and a background colour.

1. **Given the uniformly coloured class**, we notice that the colour of the background and foreground are almost identical.

2. **The coloured lip class** has a large colour contrast between background and foreground combined with a few foreground blobs. Location of the foreground blob is at the bottom centre of the flower.

3. **The striped pattern class** has a large colour contrast between background and foreground combined with a large number of foreground blobs. The flower contains strong radial edges.

4. **The spotted pattern class** has a large colour contrast between background and foreground combined with a few foreground blobs. The foreground blobs have a random location.

5. **The speckled pattern class** has a large colour contrast between background and foreground combined with a large number of foreground blobs. The edges are not dominantly radial.

Based on these visual characteristics of the pattern based classes, we deduced a set of measurable features which can be used to train any machine learning decision classifier. Figure 3.5 visualizes all processing steps needed for gathering the necessary object features from a single input image.

We start by converting the image to the La*b* colour space. This is followed by several processing steps which lead to the specific flower features used for classification:

1. A K-means clustering (K=2) is applied to all flower pixels *(after the flower segmentation step)* in the a*b* channels in order to define the dominant foreground and background colour of the flower. Each cluster is assigned with the average cluster colour. The cluster with the most pixels is classified as the dominant background colour. The first useful feature is obtained by calculating the colour difference between the two clusters (Fig. 3.5(b)).

2. The second classification feature is calculated as the relative y-position of the centre of gravity of the foreground pixels in relation to the patch.

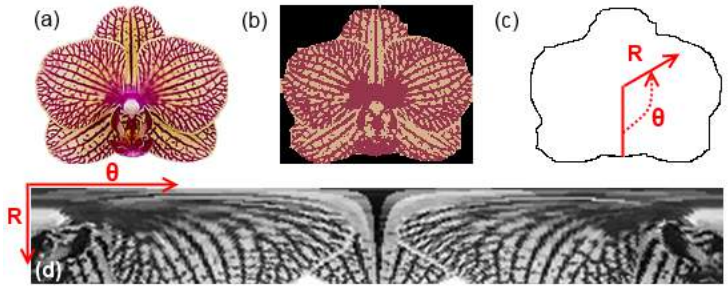


Figure 3.5: Different preprocessing steps needed for the feature calculation of each input window. (a) the segmented flower (b) K-means clustering (c/d) radial unwarping to highlight radial edges.

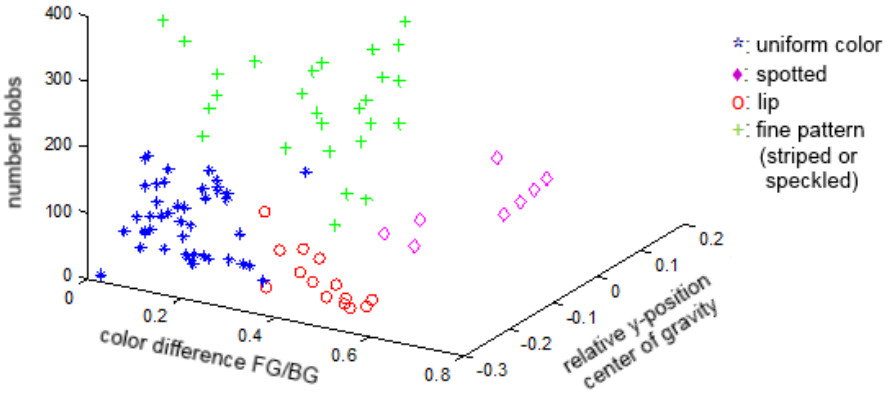


Figure 3.6: The feature space visualized with training samples of all classes.

3. A connected component analysis is applied on the binary clustered image. The ratio between the amount of blobs in foreground and background is stored as the third classification feature.

4. The radial dominant edges are quantified by applying a radial unwarping of the input image around its centre of mass (Fig. 3.5(c) and 3.5(d)). The ratio of the vertical and horizontal response to a corresponding Sobel filter is stored as the fourth classification feature. Using the centre of mass can be dangerous, since it depends heavily on the quality of the flower segmentation, so investigating a more robust solution for the centre localisation might be worth investigating.

Binary support vector machine tree

After calculating the classification features on the training images dataset, we learn a set of binary SVM classifiers with linear kernels. The use of other kernel types was impossible due to the limited size of the dataset available which would result in a very weak performing classifier.

Figure 3.6 shows the distribution of the feature space for the different flower classes that we want to separate by training a set of linear SVM classifiers. Based on this distribution, a binary decision tree based on four linear SVM classifiers was built as seen in Figure 3.7. The obtained tree structure uses a set of intermediate classes and subclasses *(uniformly coloured, pattern, coarse pattern and fine pattern)* to store in between results.

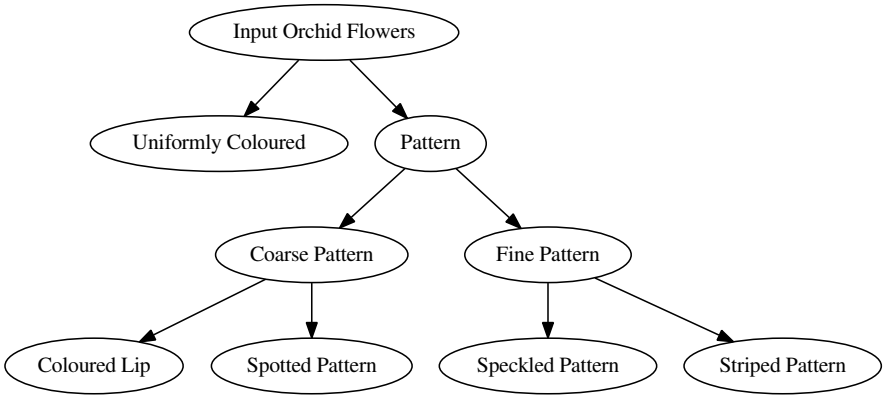


Figure 3.7: Scheme of the binary SVM classifier tree.

The complete binary tree of linear SVM classifiers is trained on a set of 97 training images of different orchid flower species, of which all four features are calculated and used for training the different binary SVM classifiers.

Classification results

After producing the complete classification pipeline, a validation set of orchid flowers was gathered containing 115 flower images of different species. The classification results on this validation set can be seen in Table 3.1. Keep in mind that the classification results are based on a single flower image level.

Overall the results are very good regarding the limited data used and the great variability of the object. However, we noticed that due to the large intra-class variance in appearance, obtaining a 100% correct classification result for each flower image will be near to impossible, especially when increasing the test set. Even domain-specific experts have problems of dividing all flowers into the correct ground truth classes *(see Figure 3.8)*.

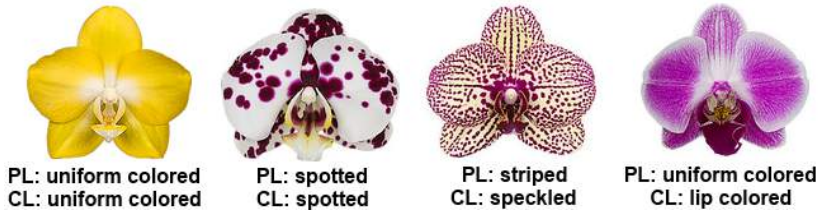


Figure 3.8: Professional (PL) versus classifier (CL) labels.

Table 3.1: Classification result on the limited validation set.

Class	Amount	Correct
Uniformly Coloured	51	94.23%
Coloured Lip	16	93.75%
Spotted Pattern	10	100%
Speckled Pattern	16	100%
Striped Pattern	23	78.26%

Detection and classification combined

Besides the neat testing images used, which were very constrained *(white background, orchids in a uniform position, single flowers, . . .)*, we also performed tests on real images retrieved from our industrial set-up using the orchid flower detector, as previously shown in Figure 3.4. The cleaned-up and segmented-out detections ensure that background clutter does not influence the feature calculation process.

We acknowledge that there are two specific cases where our simple flower-background segmentation approach yields no clean result. The first case is where the background of a detected flower is a combination of multiple other flowers. However since all the flowers of a single orchid plant have similar colours, this doesn't influence the feature calculation drastically. Secondly when flowers are too tilted and the viewpoint changes too much, then the classification process can yield wrong results. We try to avoid these cases by only training the orchid flower detection model with flower samples that have the desired viewpoint and by increasing the detection threshold so that it is high enough to simply ensure that the correct view is returned in most detections.

In all cases where the classification goes wrong for single flower instances, a majority voting to get a plant-based classification solves the problem.

Table 3.2: Classification result on complete orchid plants using the majority voting principle, to cope with single flowers triggering a wrong classification. *{# = number of detections | UC = uniformly coloured | LC = lip coloured | Spo = spotted pattern| Spe = speckled pattern | Str = striped pattern}*

Orchid	Manual	#	UC	L	Spo	Spe	Str	Majority
1	Striped	10	1	0	0	0	9	Striped
2	Uniform	11	11	0	0	0	0	Uniform
3	Speckled	16	1	0	5	3	7	Speckled

Table 3.2 illustrates this process of majority voting by supplying 3 orchid plants to our complete pipeline. First, each plant yields several orchid flowers, by performing a multi-scale detection using our pre-trained orchid model, which are then segmented out of the background as suggested previously and passed to the classification pipeline. By using majority voting each plant is labelled the correct class.

3.1.4 Conclusions on orchid flower detection and classification

In this section, we investigated the possibility of accurately detecting orchid flowers and classifying them into five larger visual texture classes. For the detection, we suggest using a Viola and Jones based approach with LBP features and AdaBoost learning to learn a boosted cascade of weak classifiers. The classification part is solved by training a binary tree of linear SVM classifiers on a limited set of training data. We conclude that we successfully built a robust orchid flower detection and classification pipeline that reaches the desired accuracy. By smartly combining the classification output using a majority voting system we achieve a perfect classification rate on the level of a single orchid plant. Knowing that we had a very limited training data set for both the detection model training and the training of the SVM classifiers, the end result is quite impressive compared to the current research in object detection, where still multiple thousands of training samples are used to reach robust classifiers. An explanation for this can be found in the controlled lighting conditions and the known set-up *(camera position, possible background, orchid flower position)*.

Given the topic of this dissertation, this research is a clear example of how scene- and application-specific constraints can be used for efficiently improving a basic boosted cascade classifier trained on only a limited set of manually annotated training samples. We clearly show that controlling the environment in which the detector is being used, results in a very robust case-specific detector.

3.2 Automated walking aid detector based on indoor video recordings

In this section, we discuss a second industrially relevant example of applying scene-constraints to an object detection problem with the goal of obtaining a robust object detector with very limited training data.

As more older adults will require medical care in the coming years, the development of automated home-care systems has evolved into an important research field. Automated home-care systems aim to automatically monitor the health of senior citizens in their own living environment, enabling the detection of an initial decline in health and functional ability and providing an opportunity for early interventions. Fall prevention is one of the research fields in which automated home care systems can be an important asset. Given that approximately one in three older than 65 falls each year [73, 113] and 20-30% of those who fall sustain moderate to severe injuries, it is clear that fall prevention strategies should be put in place to reduce this number. When these automated systems detect an elevated fall risk, preventive measures can be taken to reduce this risk, e.g. installing an exercise program to enhance gait and mobility, adapting the medication regime and introducing walking aids such as walkers.

In order to continuously monitor the fall risk of a person in their home environment, the development of an automated fall risk assessment tool is needed. For this purpose, a camera-based system was installed in the homes of three senior citizens during periods varying from eight to twelve weeks, all three using walking aids to move around. The resulting video data was used to monitor predefined trajectories using the transfer times as an indicator of the fall risk since they heavily relate to the general health of the person [54].

Figure 3.9: Input frames with a walker present and both trajectories visualized.

Previous studies [5] observed that the gait speed of a person differs when using walking aids like a walker or a cane. Transfer times measured with aids can therefore not be compared to transfer times measured without them. This underlines the necessity to automatically differentiate between the different video sequences and to determine whether a walking aid was used or not. Our presented system will focus on automatically detecting which walking aid is being used in the selected transfers based on the given video data, and more specifically focusing on the case of a walker. Examples of such a walker walking aid can be seen in Figure 3.9.

The applications of walking aid detection are not limited to monitoring these transfer times, but can be expanded to every situation where we want to have an objective measure of the indoor usage of any walking aid, like e.g. in an elderly home, where we could monitor how frequently people leave their room with a walking aid, by pointing the camera towards the entrance of the room.

In this section, we develop state-of-the-art object detection techniques to automatically detect if a walker has been used in a certain indoor walking track. The output generated by our software can in this context be used efficiently to differentiate between walker based sequences and non-walker based sequences *(using a walking cane or no walking aid at all)*. This benefits the automated analysis of transfer times, removing the need to manually label each sequence and making the measurements more meaningful by automatically adding a label of the used aid. This switches the process from semi-automatic towards fully-automated fall risk assessment. Similar steps can be applied to other walking aids like a walking cane to further differentiate the non-walker based sequences. As a result, we are able to give nurses a measure of the walker usage.

The case of detecting walking aids for elderly people is an example of an object detection application in a constrained scenery and is, therefore, an ideal case to prove the theorem suggested in this chapter, namely that scene-specific conditions can be used as constraints on top of the actual detection application. First of all, we have a fixed camera set-up, with multiple cameras observing the scene. Secondly we know that many elderly people have a fixed home set-up *(location of bed, table, wardrobes, etc.)* resulting in a known environment. This gives us two advantages, a known background and a known object size range. By using this application- and scene-specific knowledge, we succeed in building a robust walker detector. We aim at using a limited training data set and a reduced training time in order to ensure that a specific set-up can be learned within a single day timespan.

The remainder of this section is organized as follows. Subsection 3.2.1 will present related research. Then subsection 3.2.2 discusses how the data was collected and how the object model was trained using scene- and application-specific constraints. This is followed by subsection 3.2.3 in which the complete proposed processing pipeline is evaluated. Subsection 3.2.4 elaborates on the obtained output of the algorithm on real-life data. Finally, subsection 3.2.5 draws some conclusions.

3.2.1 Related work on walking aid detection

Automatic detection of a walker in random video material is not an easy task. Hagler et al. discuss that many existing behaviour measuring systems have inconvenience and obtrusiveness as a major downside [42]. Combined with that, many of the existing approaches are tested in lab environment situations. These are not representative for actual home-care situations, like in our case. The major advantages of our camera-based approach are therefore the unobtrusiveness of the system and the fact that the algorithm works on a real-life dataset.

In our application, the variance of the object, a walker, is amongst others heavily due to lighting changes, day and night conditions and different viewpoints *(fixed camera set-up but side and front view of the walker in single sequence)*. This makes it nearly impossible to use segmentation approaches like proposed by Lee et al. [59]. On the contrary, we have a known camera position, a rather known environment and a set of known trajectories that are followed by the elderly person. We prove that these restrictions can effectively limit the amount of training data needed and increase the accuracy of our detector model to meet the desired standards.

3.2.2 The setup

In this subsection, we discuss how the dataset for training the walker detection model was acquired and filtered in order to reach usable sequences for object model training. We will also take a closer look at how scene-specific knowledge was used to further improve the detection result.

The acquired dataset

The dataset is acquired using a multiple wall-mounted IP-camera set-up in the home environment of a participant recruited through convenience sampling. The participant is a seventy-five year old female living in a service flat, alternating between the use of a walker, a cane or no walking aid.

She is monitored for a period of twelve weeks in which 444 walking sequences are automatically detected and timed using the research presented by Baldewijns et al. in [5].

In this research, we only train a walker detector. We split all walking sequences into direction specific parts, given the viewpoint changes too drastically during the complete sequences to be able to train a single object model. In large parts of the recorded sequences, the walking aid is completely occluded by the user, which does not yield usable training and validation data. We define trajectory A, as seen in Fig. 3.9, which is the trajectory from the apartment's kitchen towards the table in the centre of the room, giving a more sideview of the walker. This is followed by trajectory B, from the toilet back towards the table in the centre of the room, giving a more frontal view of the walker. Since the original video sequences have double the amount of walking sequences in trajectory A, compared to trajectory B, we keep the same ratio in training and test data for both trajectories.

From the pool of available video data, we randomly grab a training set of ten sequences of trajectory A and five sequences of trajectory B. Since trajectories happen in both day and night conditions we assure that both conditions are available in the training set for both detection models. The test set contains twelve randomly picked sequences for evaluating the detector of trajectory A and five sequences for evaluating the detector of trajectory B, again both containing day and night conditions. Furthermore, we ensure no training data is included in the evaluation sets.

A manual annotation of the walker location in all training frames is performed, storing the exact location of the walking aid appearances through its bounding box location. This results in a set of positive training samples of 695 walker annotations for trajectory A and 2200 walker annotations for trajectory B. The main difference in the number of annotations is due to the duration of the sequences measured, the longer the sequence the more frames with walker appearances. As negative training data, we use the provided sequence images, since they contain all the info needed, and set the walking aid pixels to black. This results in as many negative frames without walker aids as there are positive samples. However, these images are much larger than the actual trained model size and the annotated positive training samples. Therefore we randomly sample, using the size of the model which is based on the average annotation size, 2000 negative training windows for trajectory A and 4000 negative samples for trajectory B. The ratio of positive versus negative samples is approximately $1 : 2.8$ and rounded to the nearest 1000 samples. This ratio is chosen from the experience of training multiple generic object detection models.

Since this application is used as a reference for training similar walking aid set-ups, since we are building scene and application-specific models, we also keep an eye at how long it takes to collect the training data. On average, once a person gets familiar with the annotation tool, he tends to provide around 500 annotations per hour.

Our object detection approach

We use the approach suggested by Viola & Jones in [119] as a base technique for training the object model and performing the object detection, changing the features towards local binary patterns instead of Haar-like wavelets. The main advantage of using a cascade of weak classifiers is the early rejection principle, where a small set of features is used to discard most object candidates, up to 75%. Only object candidates that pass these first weak classifiers will progress and will require more features to be calculated. This drastically decreases execution time.

For both the object model of trajectory A and B, we choose a minimum hit rate of 99.5% and a maximum false alarm rate of 50%, the default values assigned by the OpenCV implementation of boosted cascades. This means that we need to correctly classify at least 99.5% of our positive training samples at each weak classifier stage while correctly classifying at least 50% of the used negative training samples. Both models can be trained within 4 hours of processing on a basic dual core system with 8GB of RAM memory, still keeping it possible to get the whole installation set up and running within a single day timespan.

3.2.3 Complete pipeline

In this subsection we discuss the several building blocks, seen in Figure 3.10, needed to supply a correct label for each video sequence, indicating if it is either a walker or non-walker *(walking cane or no walking aid)* based sequence. We also discuss the measures taken for increasing the accuracy of our detector.

Using scene-specific information

We focus on detecting walking aids in constrained scenes like manually defined walking trajectories of elderly people or specific regions that we want to monitor. Since we want to make a system that is as versatile as possible, we start by supplying the user with the ability to define a region of interest. This mask contains the application specific regions of the input images in which the lowest point of the detection box of the walker can occur, annotated as a region on top of the ground floor and door openings. Figure 3.11 clearly shows how the user is asked to visually assign a mask to the recorded sequence.

The mask allows us to simply ignore all detections that are not part of that mask with their centroid, reducing a heap of false positive detections.

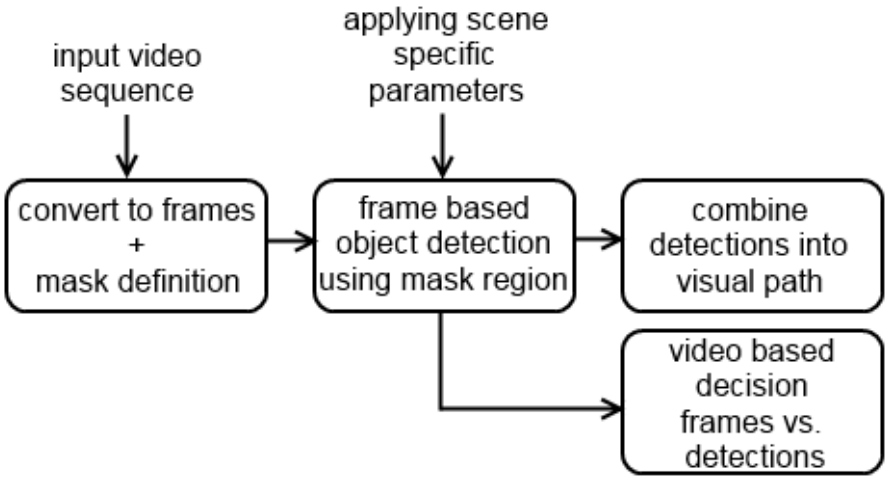


Figure 3.10: The separate building blocks for performing the video sequence classification into walker and non-walker sequences.

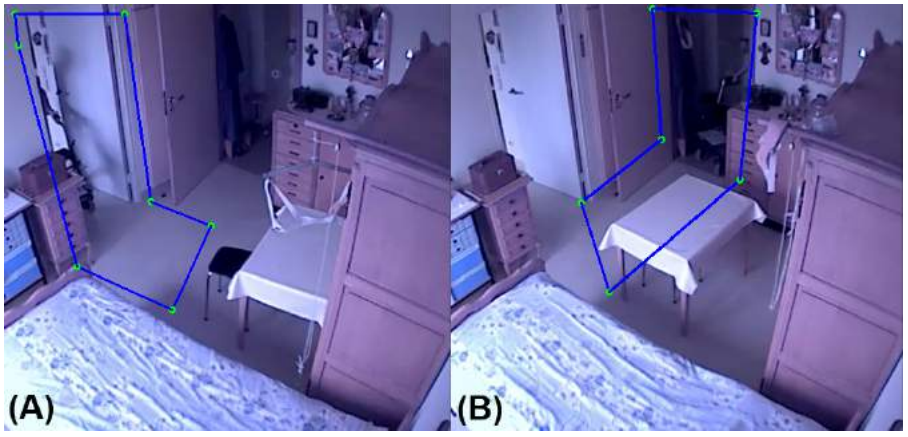


Figure 3.11: Manually defined masks for (A) front model and (B) backwards model.

Since the exact position of the camera-based capturing system is known, we can use this knowledge to apply scene-specific constraints to the detection algorithm. Based on the provided training samples we can define an average width and height of possible object instances, which are used to create a narrow scale range that prunes the image pyramid, leading to a huge reduction of object candidates in the image pyramid search space. The benefits are two-fold. First, it reduces the time needed to process a single image drastically since entire scale levels are removed. Secondly, the removal of the scale layers also reduces the number of false positive detections for a given input image. This means that the reduction of the image pyramid benefits both in the obtained accuracy and the reported processing time.

A frame-by-frame detection of object instances

Object detection is applied on a frame-by-frame basis and due to the limited training data, it still yields false positive detections *(detections triggered by image regions that do not contain an object)*. Even when there is only a single walker object in the frame, this can still result in multiple detections in that frame. Applying a predefined mask region inside the larger images reduces already the amount of false positive detections drastically, but they will never be removed completely. In order to circumvent this issue, we take a spatial relation between detections into account, by assuming only a small position shift between consecutive frame detections. This assumption in spatial relation is given by the fact that an elderly has a limited moving speed and thus that the subsequently detected walker positions should be close to each other.

The spatial relation can be found by looking for a connection between the obtained detections in the currently detection-triggering frame F_T and the selected detection in the last processed frame F_{T-1} as seen in equation (3.1). For each new detection, D_1 to D_N, of the current frame F_T, the Euclidean distance is calculated with the selected reference detection (D_R) from the last processed frame. Based on those scores, the detection with the smallest distance is kept for the current frame and stored as the new reference D_R value.

$$D_R = \min_{i=1:N} [dist(D_i, D_R)] \tag{3.1}$$

In the first frame of a sequence, the average location of all triggered detections is used, since no D_R value exist. Due to the limited mask region that is selected the error introduced by this is rather limited and can thus be neglected. Figure 3.12 illustrates this spatial relation between walker detections inside trajectory A and B and thus removing all detections that are further away. The complete calculation is repeated for each new frame where detections occur, else the frame is simply ignored in the pipeline. Applying this extra measure ensures

Figure 3.12: Detection results for (A) trajectory A and (B) trajectory B trained object models.

another large decrease in false positive detections. The combined trajectory is also visually stored for further analysis.

Performing Detection On a Sequence Basis

The visual output is good for verifying the work of the object detector on a single sequence, but when someone needs to go through a large set of sequences, then visual confirmation would be insufficient and an automated decision should be created. In our approach, we do this by training a decision classifier based on example sequences.

Due to the limited data, we cannot train a classifier that will yield no false positive detections, so simply depending on the fact that a detection occurred to classify a sequence is a bad idea. However, the number of found detections is also highly dependent on the number of frames that were processed from the initial sequence. We calculate the ratio between the number of walker detections over the complete sequence (#D) and the total number of frames (#F) inside a walking sequence. This ratio is called the walker certainty score (C_{walker}):

$$C_{walker} = \frac{\#D}{\#F} \tag{3.2}$$

We determined the optimal value of the walker certainty score by randomly selecting a subset of 20 validation sequences from the remaining video data, of which 10 walker based sequences *(half trajectory A and half trajectory B)* and 10 non-walker based sequence *(again half trajectory A and half trajectory*

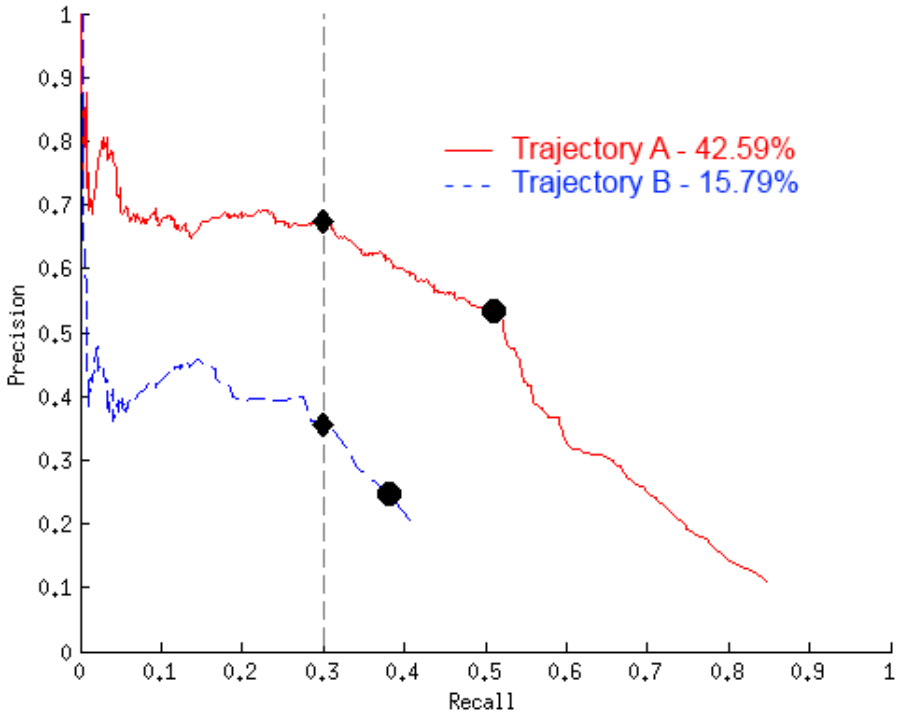


Figure 3.13: Precision-Recall curves for frame-based object detection on both trajectory A and trajectory B: • denotes theoretically best configuration, while ♦ denotes the selected configuration.

B), containing both day and night conditions and in the case of the non-walker sequences containing walking cane and no walking aid sequences. For this complete validation set, we calculated the walker certainty score using the appropriate detection model and analysed the data. We selected walker certainty threshold that was low enough to classify walker sequences with less detections still as a walker sequence but ensuring that the few false positive detections that can occur in a non-walker sequences *(e.g. by lighting changes or a walking cane that does trigger a detection)* still get discarded and thus also get correctly classified. This yields a walker certainty score threshold of 0.2, which works fine for both trained models, but which can be re-evaluated once a larger set of previously unused walker and non-walker sequences is available.

3.2.4 Quantitative detection results

Frame-by-frame detection results

Figure 3.13 shows the Precision-Recall curves for the detection models for both trajectories, together with the average precision of both models on a frame-by-frame base *(calculated by the area under the curve (AUC))*. We notice that the AUC value for trajectory A is better than for trajectory B. This can be explained by the fact that the training data for trajectory B was biased due to the walker occlusion from the dining table being in front of the trajectory.

In theory, the best operating point *(indicating the best detection threshold)* is reached by selecting the curvature point that is closest to the top right corner, denoted by •. However, in our application, we want a detection threshold that yields as few false positive detections as possible but still ensures that there are enough true positive detections *(true positive detections with lower score also disappear when increasing the threshold)* to retrieve a walker certainty score that is over 0.2. In our case this position was set at a recall level of 0.3, denoted by ⧫. This was validated on the same validation set that was used to determine the optimal walker certainty score. Ideally one would optimize both parameters together, instead of independently from each other and fixing the walker certainty score at 0.2, which might allow us to use another position on the precision-recall curve.

Sequence-based detection results

The precision-recall curves are based on a frame-by-frame analysis, the most common manner of evaluating object detection algorithms. In order to get a performance measure on our complete pipeline, we collected a mixed test set of 26 walking sequences from both the frame-by-frame test set and the validation set of the walker certainty score. For both the models of trajectory A and B, we used sequences containing walkers (16), a walking cane (6) or no walking aid (4) at all. The results of this walking sequence-based classification can be seen in the confusion matrix in Table 3.3.

Table 3.3: Confusion matrix of algorithm output using both trajectories A and B with a $C_{walker} = 0.2$.

	Walker Classified	**Non-Walker Classified**
Walker Sequence	14	2
Cane Sequence	0	6
No Aid Sequence	0	4

We do acknowledge that still 2 walker sequences *(related to trajectory B)* are classified incorrectly. After a small investigation into the matter, it seems that their threshold was just below the set 0.2 walker certainty score. Further testing and validating over a larger set to find the most optimal threshold is required to resolve this issue.

Training data versus accuracy

Since the goal is to supply a tool for caregivers that needs as little manual input as possible to obtain a result as accurate as possible, we roughly tested the relationship between the amount of training data and the accuracy achieved by the frame-by-frame walker detector, since it is the core part of our pipeline. For this we took the model of trajectory A and varied the amount of positive and negative training data *(see Fig. 3.14)*, while still keeping the pos:neg ratio at 1:2.8, which is the same ratio as for training the model.

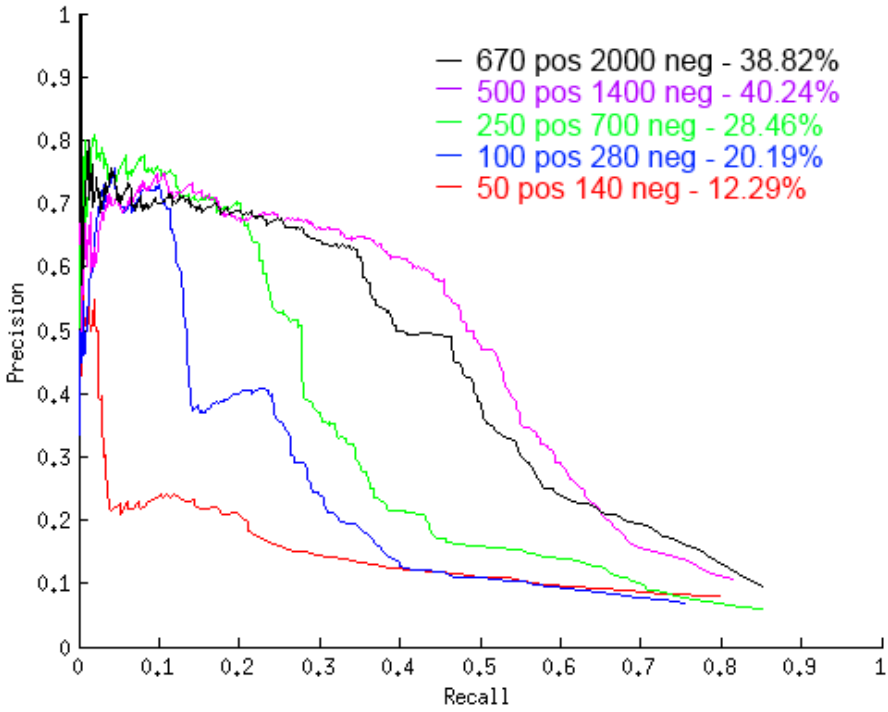


Figure 3.14: PR curves under varying amounts of training data with the achieved object detection accuracy defined by the area under the curve (AUC).

We noticed that the first models show a steep increase in accuracy when adding more data while reducing a bit when arriving at the final models. For the complexity of the set-up, this test suggests that we would have achieved a slightly better result by using only 500 positive samples instead of all the 670 positive samples. This behaviour can either be explained by the natural variation in training iterations or possibly by over-fitting on the training data *(where the average precision is dropping on a separate set of validation samples, but still increasing on the given set of training samples)*. A deeper investigation in the matter is necessary to make firm conclusions.

3.2.5 Conclusions on detecting walking aids

The goal of this section is to automatically detect walking aids in walking videos of elderly people with the goal of improving automated fall risk assessment. We applied an LBP feature-based approach with an AdaBoost learner to robustly create two object detectors for walkers in two separate trajectory directions. Several improvements to the basic algorithm of Viola and Jones based on the scene and set-up specific knowledge were applied.

We prove that our approach can be used to automatically label video sequences for automated fall risk assessment. While the detection rates we achieve on a frame-by-frame basis are only around 68% for trajectory A and 36% for trajectory B *(both measured on a recall of 0.3)*, we achieved 92.3% accurate detection results on full trajectory video sequences by adding the spatial relation constraint and the walker certainty score thresholding. Secondly, we show that we create an easy to train system given a specific walker type, of course for a specific situation, but limiting the training time to just a couple of hours while still preserving decent accuracy. This ensures that it is easy to set-up a new system for another walker type in another location. We also prove that the amount of training data can be related to the achieved detection accuracy and the set-up complexity. Since the performance difference between the best model and the actually used model for this approach is rather limited, we ignored the influence on the sequence-based validation and did not repeat all tests.

For validating the complete pipeline with the case of using a walking cane, the available training set was too limited and the cane itself existed of only a few image pixels due to the low camera resolution, which does not work well with the object detection framework suggested by Viola & Jones.

In this section, we proved that using scene- and application-specific constraints clearly helps for building and improving basic object detectors. Furthermore, we prove that we do not always need a highly accurate detector, as long as our complete pipeline scores accurately enough given the application-specific task.

3.3 Safeguarding privacy by reliable automatic blurring of faces in mobile mapping images

This final section of chapter 3 discusses a third industrially-relevant application, where using scene-specific constraints helps to improve the object detector output. It does so by transforming the constraints into smart post-processing filters, allowing us to initially use a detector with a high recall and a lower precision. By then applying the post-filters, we reduce the false positive detections and thus increase the precision.

For mobile mapping applications, a vehicle equipped with cameras is used to capture images in order to give the user a digital view of the surroundings. This is repeated at pre-set intervals in order to ensure that the complete surroundings of the car are being captured. Companies like Google, but also local land surveying offices, are carrying out such measuring campaigns to make digital images of streets across the globe. When collecting all this data, one can imagine that the amount of data increases drastically once someone is capturing larger projects, e.g. the 'Google Street View' application. The goal of capturing all this data is providing users with fast, accurate and detailed data measurements for producing all kinds of 2D and 3D geographical information systems.

Avoiding pedestrians, walking around the scene when capturing mobile data, is nearly impossible, which raises the question of privacy issues when they are actually recorded. Especially when this data is shared with or sold to industrial partners, it is important that the privacy of these pedestrians is guaranteed. Therefore companies are continuously looking for robust solutions able to filter out privacy-sensitive content from the captured data.

One solution could be to manually browse the data, indicating every pedestrian and making them privacy-safe by applying a blurring filter. In the case of limited data, this might be the fastest and most accurate solution. However when the data size rises over several millions of images a week, one immediately notices that this approach is no longer suitable. In those cases an automated approach is preferred, like applying pedestrian detection algorithms [16, 119, 26, 31] on the captured mobile mapping data, marking possible pedestrian-like areas in the image. These, in turn, can then be blurred or cut out, to avoid transferring privacy-sensitive data.

A major downside of existing pedestrian detectors is that they require the manual selection of a threshold on the detection certainty score to find a good balance between finding actual pedestrians in the image and avoiding false positive detections. If the threshold is set too strict, we will only detect pedestrians but

be unable to find all of them, and thus privacy issues arise again. If we put the threshold too sloppy, all pedestrians will be found, but similar objects or areas will trigger a false positive detection which will also be blurred. We want to avoid this at all costs because data is used to derive GIS systems, which need to be as accurate as possible.

In this section, we propose an effective post-filtering step using scene-specific constraints, by setting a sloppy detection certainty threshold, avoiding false negative detections *(missed pedestrians)*, but additionally ensuring the removal of false positive detections using several post-processing steps. Furthermore, we expand the system with additional small colour-based classifiers able to remove even more false positives. Finally, we provide an elegant soft blurring approach for safeguarding the privacy of pedestrians inside mobile mapping images.

The remainder of this section is structured as follows. Subsection 3.3.1 presents related research, while subsection 3.3.2 discusses the data collection. This is followed by subsection 3.3.3 in which the proposed approach is discussed in detail. Finally, subsection 3.3.4 elaborates on the obtained results while subsection 3.3.5 sums up conclusions.

3.3.1 Related work on scene-constrained pedestrian detection

Van Beeck et al. [117] introduced a warping window approach where fast real-time vision-based pedestrian detection is obtained by calibrating the height and orientation of the pedestrian at each specific image location. We prove that we can apply a similar technique, as a post-processing step after the detection phase, by learning a relationship between the height and position of an average pedestrian from a limited set of application-specific annotations. We explicitly decided to use this as a post-processing step, to ensure we have as many candidates as possible, which can then be pruned by our filters. Furthermore, our filters are based on the output of the detector, which is not available before training.

Earlier in this dissertation, we proved that using application-specific information, is one way to improve the accuracy of object detection algorithms. Similar rules apply for pedestrian detection, as far as the application allows you to find some application-specific constraints. In our application we exploit the fact that the camera is mounted on top of a car, at a fixed position with respect to the ground plane, resulting in a relation between the position and the height of any given pedestrian. Furthermore, we exploit the annotated training data to learn regions of interest, avoiding the processing of undesired image regions, like the sky or on top of buildings. Several papers [14, 84, 22] describe the similar

use of a ground plane assumption for 3D modelling and multiple camera view processing.

For privacy masking, several solutions have been proposed. Tanaka et al. [111] try to define how much blurring is needed to reach a certain level of privacy. Ilia et al. [82] apply a simple block based blurring, whereas Nakashima et al. [76] suggest using image melding, replacing a person's face with a fixed neutral expression instead of blurring. Our application still demands masking, but avoiding the hardness of block-based blurring, we propose to use a smooth soft-blurring approach.

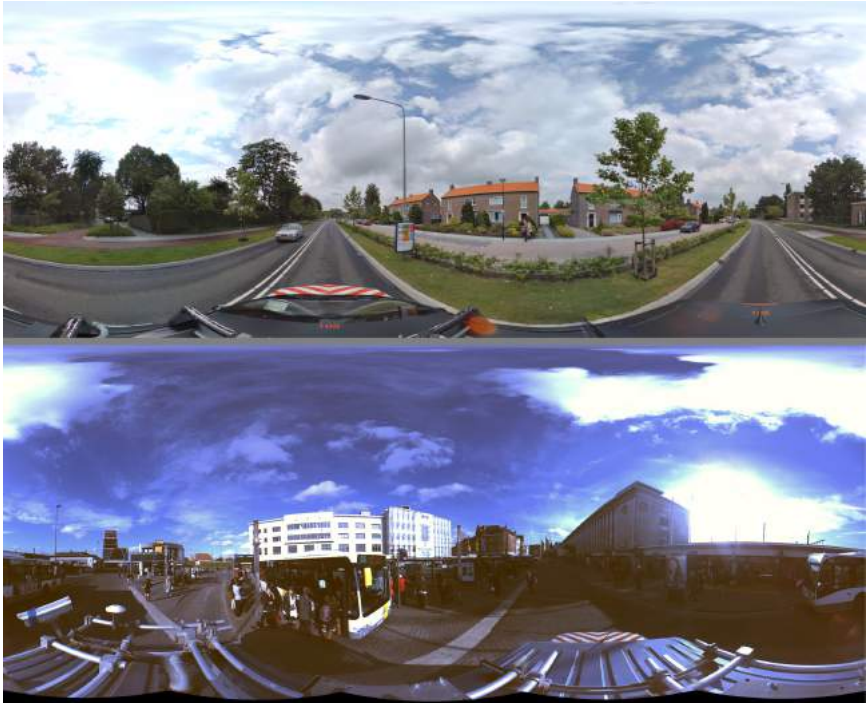


Figure 3.15: Example frames for both datasets used: (top) Dataset 1 - An urban area in the Netherlands; (bottom) Dataset 2 - Belgian train and bus station.

3.3.2 Datasets

This research is developed on top of two mobile mapping datasets, which are made publicly available[1], to encourage further research in this area. An example of both mobile mapping datasets can be seen in Figure 3.15.

The first dataset is a series of mobile mapping cycloramic images with a resolution of 4.800×2.400 pixels, captured using a LadyBug 1 camera set-up, in a quiet and calm urban area in the Netherlands. The captured images give a full 360-degree view from the surroundings of the car at any given position. The camera itself is fixed and mounted on the top of the roof of the car. The set has 450 images under daylight conditions. The dataset is used to develop and fine-tune the used approach.

The second dataset was captured using a LadyBug 2 camera, having a resolution of 8.000×4.000 pixels, again mounted on top of the roof of the car, containing 45 images of a train and bus station in Belgium. We used this dataset to prove that the developed approach is independent of the application-specific settings like camera set-up and application environment, except for defining the actual height-position relation used to improve the detection success rate.

All database images were manually annotated to provide ground truth data for the actual locations of pedestrians. For the first dataset, this led to 240 pedestrian annotations while the second dataset contained 1630 pedestrian annotations. The large difference is mainly due to the recorded surroundings, where a train and bus station is likely to have more pedestrians walking around in each mobile mapping image.

3.3.3 Used approach

The used approach can be split into several processing blocks, as seen in Figure 3.16. We start by creating a limited amount of ground truth annotations for both datasets, needed for both building the height-position relation used as application-specific constraint and inferring the colour-specific constraints for learning the false positive elimination classifiers. The annotations are also used for validating each additional post-processing step. This block is seen as the offline processing part since it does not need manual interference except the initial annotations.

At runtime, the online processing part, we apply a multi-scale pedestrian detection algorithm on the input image, storing the detection results together

[1] http://www.eavise.be/MobileMappingDataset

with their detection probability. We prune the detections using our scene-specific constraints *(a height-location relation and a certainty score thresholding)*, resulting in an allowable region reduction. For specific object classes that still trigger false positive detections, we design specific colour-based false positive elimination classifiers *(modelled as weak Naive Bayes classifiers)*, which in turn further improve the average precision.

The main goal is thus to remove as many false positives as possible and to increase the resulting average precision of our algorithm. The final retrieved detections are then parsed by a soft-blurring filter to ensure privacy safeguarding, by removing features that can help to recognize the person detected.

As pedestrian detector in this research, we use an open-source C++ implementation of the cascaded Felzenszwalb latentSVM4 implementation [20], which uses a part-based object detector for efficient pedestrian detection. The reason for this is quite straightforward. A part-based detector is non-rigid and thus captures the different poses of pedestrians more efficiently than a rigid pedestrian detector. On the other hand we have a fast and optimized C++ implementation available, that is easily integratable in our pipeline. However, our post-processing step is independent of the pedestrian detector used, so basically it can be replaced by any out-of-the-box pedestrian detector. This is one of the major benefits of our system, encouraging cross-dataset and cross-algorithm evaluation.

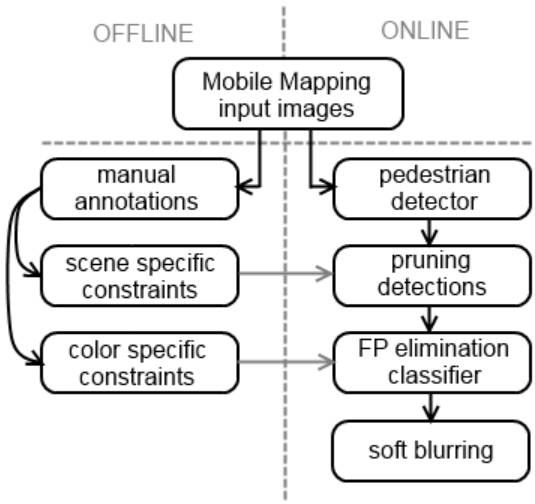


Figure 3.16: Block diagram of the used approach.

A major downside of every pedestrian detector is that one must filter the detector output based on the object detection probability score, by selecting a probability threshold and thus locking down on a specific point on the precision-recall curve of that detection model. If we decide to put the threshold too sloppy *(a low probability threshold)*, we get an increase in false positive detections, while at the same time, reducing the number of false negative detections *(every single pedestrian will likely be returned)*.

This will lead to an enormous amount of privacy masking, also removing a lot of useful information from the image, which is unacceptable for the mobile mapping community.

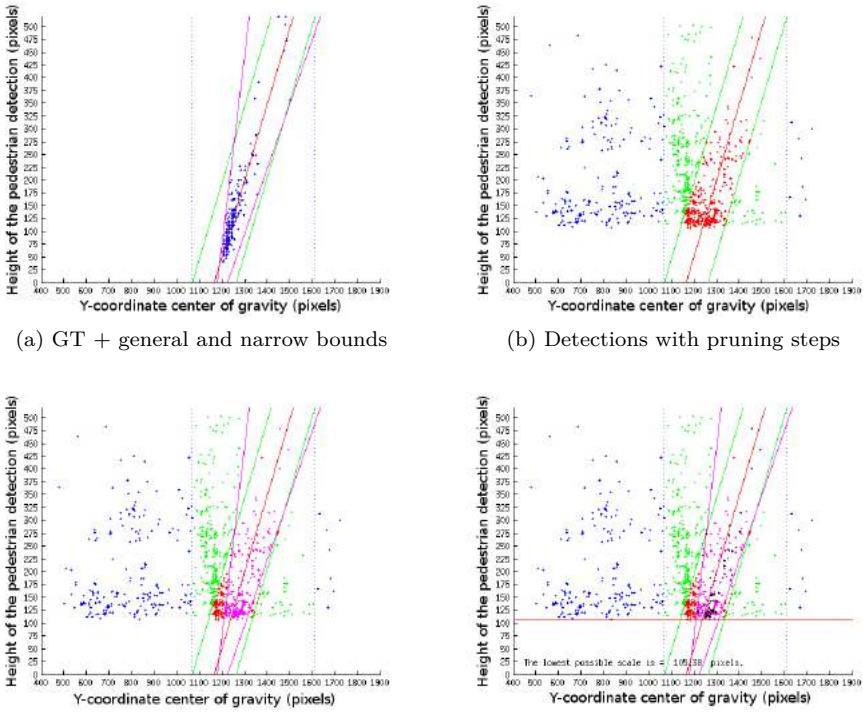
On the other hand, if we put the probability threshold very strict *(a high probability threshold)*, the amount of false positives will decrease drastically, but result in an increase in false negative detections, which in our application of privacy masking would not be acceptable. Therefore, our approach uses a sloppy threshold for obtaining every possible pedestrian as a true positive detection, then subsequently using smart post-processing to efficiently remove as many false positive detections as possible.

Scale-space location relation

When considering our application of mobile mapping using a fixed 360-degree cycloramic camera, we know that the actual height position of the camera, compared to the environment, will be fixed, only if we assume a flat ground plane, and if that ground plane will never change drastically *(small bumps in a road will not generate issues)*. This is a crucial scene constraint, allowing us to take into account that every pedestrian in the image, walking on the street or on the sidewalks, has an average fixed height in relation to the position in the final cycloramic image. People closer to the car and thus to the camera will be larger, while people further away will move towards the camera's vantage points and thus be smaller. For any given horizontal line in the image, we can state that all pedestrians on that line will have the same average height, of course keeping in mind that we have a natural height variance within pedestrians.

Mapping ground truth annotations

In order to model a height-position relation, we start by mapping out the ground truth annotations collected on the first dataset. The effort of annotating a smaller part of application-specific data, to be used for deriving scene- and application-specific constraints, is rather small compared to training a complete new pedestrian detector *(which needs much more annotations and processing time)*. The result of these manual annotations can be seen in Figure 3.17(a), where the height of each annotation is mapped in relation to the position in the given cyclorama, defined as the centre of mass.



(a) GT + general and narrow bounds

(b) Detections with pruning steps

(c) Applying narrower bounds

(d) Score threshold

Figure 3.17: The height-position relation building process.

During the annotation phase, only pedestrians on sidewalks, parks and roads were annotated.

Model fitting and region reduction

We fit a linear model to the data points in Figure 3.17(a) and search for image region boundaries. The red curve is the fitted linear relation to the mapped annotation data, which models the relation between pedestrian height and pedestrian position, relative to the camera position. The green borders are based on the assumption that we have a Gaussian data distribution compared to the fitted model, and that these borders should capture 99.8% of all detections using the rule of $[-3\sigma, +3\sigma]$. The reasons for allowing this deviation are quite straightforward. First of all, we have a natural deviation in pedestrian height, while secondly, due to the car's suspension, the camera height is not completely fixed to the ground plane. Thirdly, there is a possible deviation from the flat

Figure 3.18: Applying borders to minimal and maximal pedestrian height, as defined by the blue borders in 3.17(a).

ground plane assumption caused by height differences due to sidewalks, defects in the road, speed bumps, etc. The blue dotted borders define allowed position regions for pedestrians in the image, assuming the training data covers a wide variance of available pedestrians specific to the application. This is visualized in Figure 3.18 and allows us to immediately ignore detections that are outside these regions, removing about 50% of the image and thus lowering the chance of false positive detections occurring.

Applying constraints on detection data

Figure 3.17(b) visualizes all detections obtained by our pedestrian detection algorithm. By applying the realistic pedestrian occurring boundaries, we obtain the green dots, representing pedestrian detections in reasonable and allowed positions in the image. We notice that this allows dropping a significant amount of false positive detections already. Subsequently, we apply the rough deviation borders *(green lines)* on top of the green data points, demanding that our detections also fit the height-position relation. This in return removes a large part of false positive detections, keeping only the red detections as acceptable pedestrian detections.

Updating the distribution constraint

We acknowledge that assuming a Gaussian distribution around the fitted linear height-position relation might not always be the best choice, especially when you consider the fact that when moving further from the car, differences in pedestrian height become less obvious to notice, certainly at pixel level, whereas close to the car height differences are clearly visible. Therefore we updated the green borders, to closely map the correct distribution of the annotation data,

which can be seen in Figure 3.17(a) as the magenta borders. Applying these on the actual detection output, again removes several false positive detections, resulting in the magenta coloured detections seen in Figure 3.17(c).

Detection certainty score thresholding

We decided initially to put the detection certainty threshold very sloppy, to ensure that the amount of false negative detections is close to zero. Now that we automatically removed multiple false positive detections, we can look back to this setting and adapt it to our application-specific needs. Due to less cluttered images filled with detections *(most false positives are removed now)*, it becomes easier to visually select a decent score threshold for our application. When using pedestrian detectors in the wild, we observed that the used LatentSVM4 detector almost never returns valid pedestrian detections when the certainty score is below zero. Of course, this value is application specific and can change drastically when considering other application fields. In our application, detections with lower scores mainly resemble objects that have similar feature descriptions, like a small tree or a traffic sign, but in 99% of the cases, they do not match actual pedestrians. Since we want to avoid blurring too much valuable image information, we enforce an extra pruning rule, demanding a detection certainty score equal to or above zero. This results in the black detections *(see Figure 3.17(d))*.

Visually verifying the filtered detections

When visually checking the data, we wonder why very small pedestrians in the background were ignored by the pedestrian detection interface. As seen in Figure 3.17(d) we calculate the smallest retrieved detection height by the DPM detector, which is a height of 105 pixels. Considering this in relation to the pre-trained pedestrian model, this is actually normal, because the model is trained with a fixed training sample height of 124 pixels, keeping a small area of background around the pedestrian, also called padding. At detection time, the model dimensions always limit the smallest possible detection height, so if we would like to include these smaller pedestrians, we should first up-scale the input data. However, we should keep in mind that this introduces image artefacts which could interfere with the pedestrian detector performance itself. In our application, this is not a real problem since pedestrians with a height below 100 pixels are already privacy secure and impossible to recognize when looking at the complete mobile mapping image of 8.000×4.000 pixels [111].

Colour-based removal of pedestrian-like detections

Even with all the proposed post-filtering steps applied, we noticed that some object classes continuously succeeded in triggering false positive detections. Take for example the case of small traffic signs indicating the traffic flow when

Figure 3.19: Example of the need for an extra filter for traffic signs still passing the post-processing steps.

entering a roundabout, as seen in Figure 3.19. As humans, we clearly see the difference between a pedestrian and this rigid traffic sign. However, due to the specific nature of pedestrian detection algorithms, it is normal that these false positive detections occur. First of all, the used DPM algorithm by Felzenszwalb et al. [31] ignores colour information, since the variety of colour in pedestrians is enormous. Secondly, as a feature it uses edge information of deformable parts. And this is exactly where the biggest problem occurs. The mentioned traffic sign has a top part that is very similar to a head and a middle and bottom part that have similar feature responses as a human body. Since the body and the head are parts with a big weight in part-based pedestrian models, it is important to add an extra pruning step to remove these false positive detections that are still classified as valid detections by our pipeline. Especially in the context of mobile mapping, it is important that crucial road information is not blurred out due to privacy reasons, since clients interested in this data are looking for exact locations of traffic signs, e.g. to keep an automated index of road signs.

To avoid this kind of problems we propose a simple pruning step using a simple Naive Bayes classifier. This machine learning technique takes a limited set of positive and negative training samples and, based on some calculated features *(e.g. very simple colour-based features)* from the training data, decides whether a valid detection should still be classified as pedestrian or not. We prefer using a machine learning approach towards setting hard thresholds on basic features because it is more robust in finding the optimal separation between classes once more training data is supplied. As seen in Figure 3.20, we use a very small positive and negative training set *(both containing only five samples)*, where we tried to include as much traffic sign like pedestrians in the negative set as possible *(by looking for matching colours)*, to avoid that those would now get filtered out, e.g. when someone is wearing a bright jacket. Finally, we constructed a small test set to evaluate the success rate of our classifier.

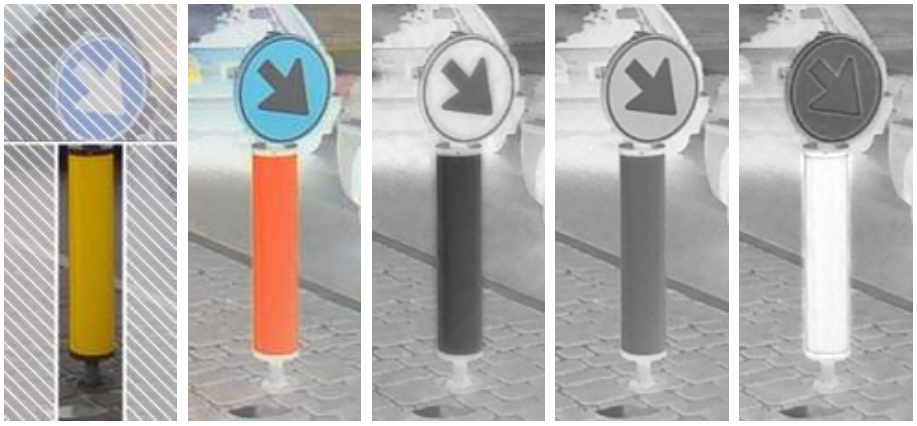
(a) Positive training set

(b) Negative training set

(c) Test set + Classification result (green = sign / red = pedestrian)

Figure 3.20: Positive, negative training and test set for traffic sign filtering.

From each training sample, a set of simple visual features are calculated. In this case, the most distinct feature is the bright yellow colour of the 'body' part of the traffic sign. We separate the top 30% of the image and then split up the bottom 70% in 3 equal regions, as seen in Figure 3.21(a). The middle



(a) (b) (c) (d) (e)

Figure 3.21: Naive Bayes features: (a) original (b) CMY(K) (c) C response (d) M response (e) Y response.

area is then transferred to the CMYK color space *(Figure 3.21(b))* because the traffic sign has a very low response in the C layer *(Figure 3.21(c))*, an average response in the M layer *(Figure 3.21(d))* and a high response in the Y layer *(Figure 3.21(e))*. This behaviour is not equal for pedestrians. We take the average CMY values for this smaller window and use that as a feature vector for each positive and negative sample. The K channel is simply ignored.

Finally, when running the classifier on the provided test set, the samples were all classified correctly either as a pedestrian or as a traffic sign and thus the simple classifier proved to work as an effective post-filtering step. Similar behaviour was detected for specific kind of bushes, so again an extra Naive Bayes classifier could be constructed for this purpose. The advantage during post-processing is the execution time of these extra filters is computationally very cheap (~1ms) due to very simple features used.

Soft-blurring approach

The final step of our proposed pipeline is to obtain the valid detected pedestrian regions and apply a local privacy-safeguarding filter to them. In agreement with some mobile mapping companies, we provided the option for both pedestrian and face region blurring. An intuitive way to apply privacy-safeguarding would be to apply a standard Gaussian blurring filter. One of the main downsides to this is the existence of very prominent edge artefacts which cannot be removed, as seen in the left part of Figure 3.22. We would prefer a blurring filter that is

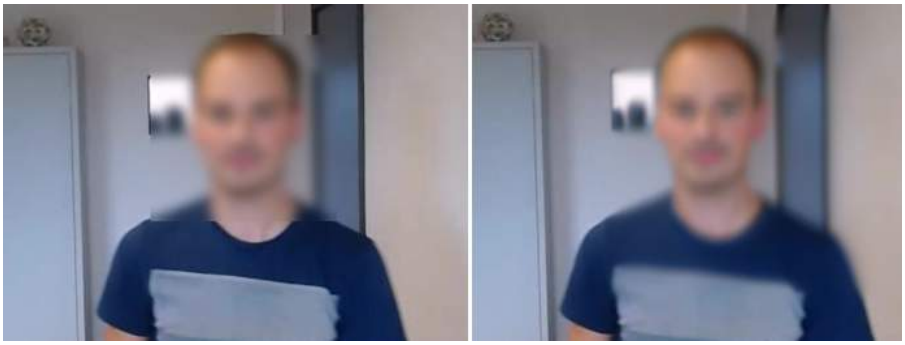


Figure 3.22: Blurring filters (left) standard Gaussian blur (right) smooth blurring filter.

not as strong on the edges, as seen in the right part of Figure 3.22, but which is strong in the middle and softens up towards the edges of a detection. This ensures privacy but the end result is visually more pleasing.

Instead of convolving the image region with a Gaussian kernel with a fixed size and sigma, we propose a convolution with an adaptable Gaussian kernel, where the sigma (σ_{kernel}) is defined as a function of the normalized pixel distance Ψ to the centre of the detection itself as described in equations 3.3, 3.4 and 3.5. To ensure that the blurring is proportional for differently sized pedestrian detections, we add an extra size dependency Δ, which takes into account the area of the detection found compared to the area of the original image. This ensures that in the end, each detection is equally blurred.

$$\Psi = 1 - \frac{d(center_{detection}, position)}{r} \tag{3.3}$$

$$\Delta = \frac{area(pedestrian)}{area(image)} \tag{3.4}$$

$$\sigma_{kernel} = 0.1 + (\Delta\Psi^2) \tag{3.5}$$

We apply this soft blurring filter to every pedestrian detection in a given input image, blur out the detected pedestrians or their associated face region and make the captured mobile mapping image privacy safe. In our application, we applied face blurring which can be seen in Figure 3.23 and 3.24. This is simply passed as an extra parameter to our smooth blurring function. In order to make the blurring regions more visible, we also visualized the actual detections.

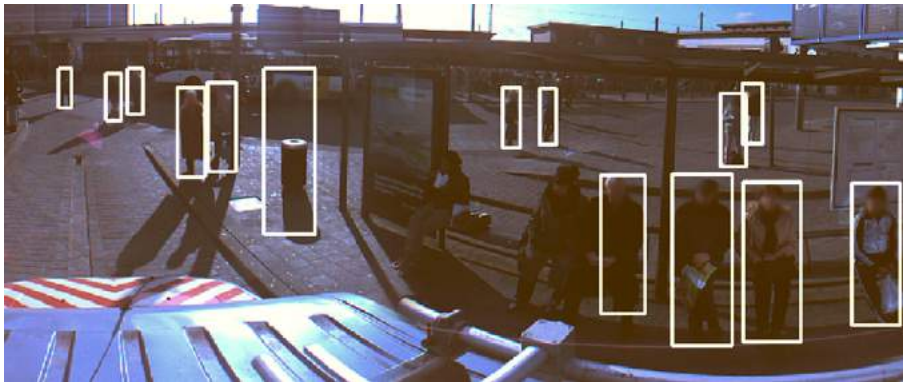


Figure 3.23: Applying face region soft blurring on filtered pedestrian detections.

Figure 3.24: Close-up of privacy smoothing using only the face region of the detection.

3.3.4 Obtained results

We applied the same post-processing steps discussed in section 3.3.3 *(enforcing valid pedestrian regions, applying a height-position relation and adding a scoring threshold)* to the second dataset and got similar improvements. The colour-based Naive Bayes classification was left out since the specific object class does not

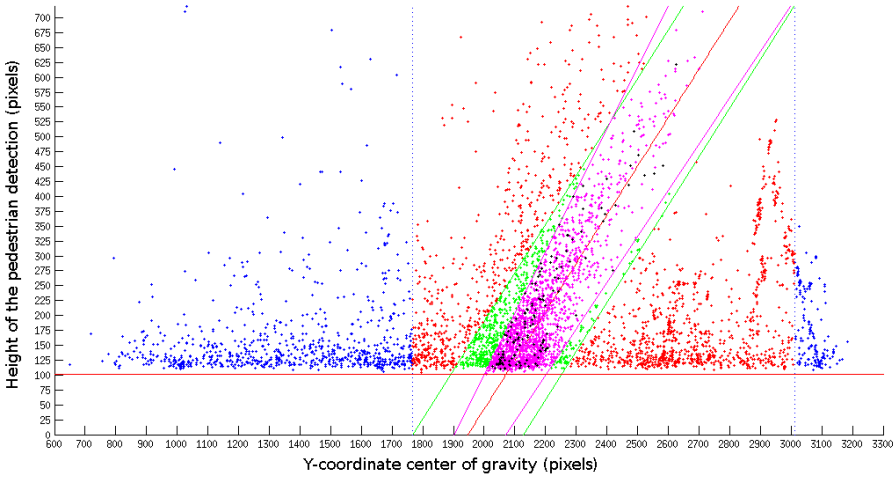


Figure 3.25: Applying all post-processing steps to the output of the LatentSVM 4 INRIA based pedestrian detector applied on dataset 2.

occur inside the dataset. The result of pruning the detection output can be seen in Figure 3.25, while visual results of applying these constraints can be seen in Figure 3.26. Especially pay attention to the false positive detections on the car that are disappearing as well as some of the double detections.

In order to make sure that we actually achieved an improvement over simply using the out-of-the-box available pedestrian detection algorithm, we evaluated the number of true positive, false positive and false negative detections after applying the different post-processing steps discussed in section 3.3.3. The result of this comparison can be seen in Table 3.4. We lowered the initial DPM certainty score threshold for this dataset to -1 to get a recall as high as possible.

Using precision-recall curves, we visualized the accuracy gained by applying the mentioned post-processing steps to the detection results on dataset 2, which can be seen in Figure 3.27. Notice that an out-of-the-box object detector already

Table 3.4: Comparison of TP, FP and FN values after each post-processing step for first dataset.

	#TP	#FP	#FN
DPM orig.	928	4159	349
After pruning	928	3182	349
After height-position	852	1015	384

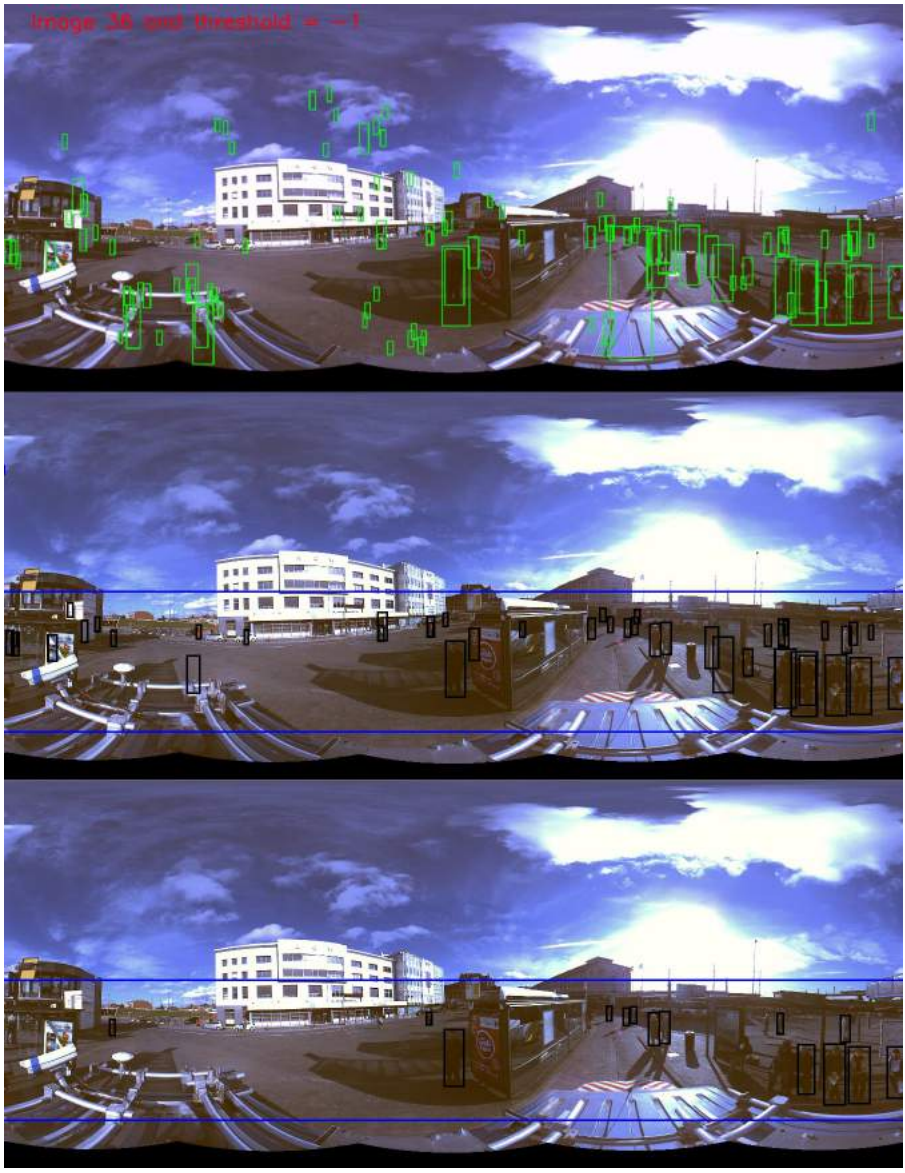


Figure 3.26: Example of applying post-processing steps to data from the second dataset. (top) original detections at score threshold -1; (middle) after pruning; (bottom) after height-position relationship and score > 0.

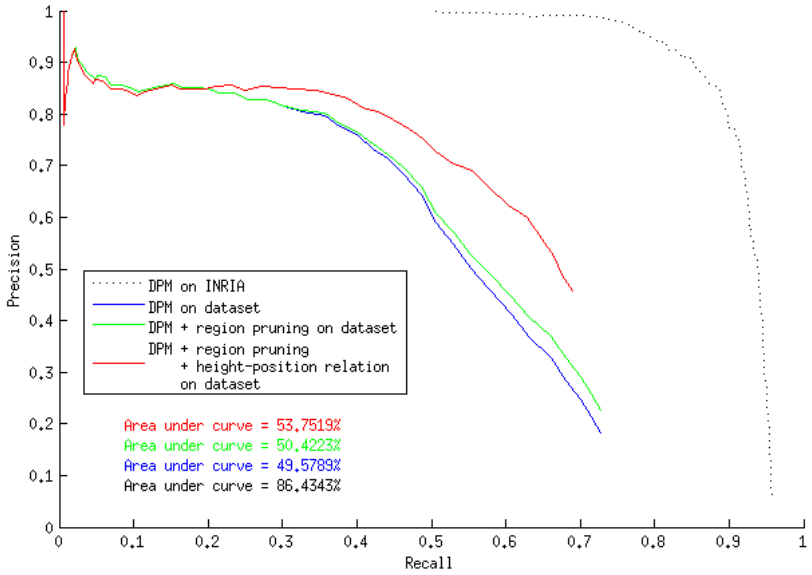


Figure 3.27: Precision-Recall curves generated for dataset 2 with all post-processing steps applied and the reported accuracy using the area under the curve measurement.

experiences a large accuracy drop when doing a cross-dataset evaluation, which is called dataset bias [115], and that there is a substantial accuracy gain when applying our post-processing steps.

Since the detection analysis for dataset 2, shown in Figure 3.25, proves that the minimum object size found by the used DPM model is 101 pixels, we ignored possible ground truth annotations on pedestrians smaller than 100 pixels, to make an as accurate precision-recall curve as possible. The remaining false negative detections are mainly due to people sitting on benches, riding bikes or motorcycles, which are less likely to get detected by the used pedestrian DPM model and which is a known issue. Solving this issue can be done by combining the pedestrian DPM model with for example a bicyclist DPM model.

We do acknowledge that this solution is far from 100% fail proof. There are still some bottlenecks that should be taken into account. The overall approach is generated to improve the output of any available pedestrian detector without retraining the actual DPM model specific to the application. However, up till now, there is not yet a single off-the-shelf pedestrian detector which is able to detect every single pedestrian out there in any given application, especially when performing cross-dataset validation [115]. While our approach focuses

mainly on improving the precision rate of our detector, as seen in Figure 3.27, getting the precision of pedestrian detectors to 100% in any given application is still a very challenging task and an actively researched topic.

3.3.5 Conclusions on privacy safeguarding

The goal of this sections is to efficiently blur pedestrians in mobile mapping images to avoid privacy-related issues while safeguarding as much image information as possible. By using an off-the-shelf pedestrian detector trained on a different dataset and setting a sloppy confidence threshold, we proved that applying efficient post-processing filters, based on application-specific constraints, e.g. a height-position relation, can greatly improve the detection outcome. In addition to the proposed height-position filtering step, we supply additional easy-to-train and lightweight Naive Bayes filters for objects that still trigger false positive detections, e.g. roundabout traffic signs, without the need of large amounts of annotated training data.

We prove that in a specific situation, we can use pre-trained pedestrian detection models, but, given a limited amount of manual annotations on a situation-specific dataset, we can boost the detection accuracy enormously by exploiting scene-specific constraints, e.g a known ground plane assumption. Finally, we propose an elegant soft-blurring alternative to a standard Gaussian blurring filter, for privacy-masking reasons, by adaptively changing the parameters of the Gaussian kernel used for the convolution with the found pedestrian detections.

Our application focuses on detecting pedestrians walking on the modelled ground plane, which raises a new problem. People standing on a balcony, sitting on a bench, lying on the grass or driving a bike, will not fit into this ground plane assumption and will thus simply be filtered out by our approach. We could improve our approach by using multiple detection models, for these different pedestrian classes and then apply separate post-filtering rules for each detector.

Given the current bottlenecks, we are still convinced that this work is valuable for people handling privacy-sensitive mobile mapping data. This research allows users to automatically remove privacy sensitive data from their captured datasets, without the need of manually handling each image *(which would be very costly and time-consuming)*. Furthermore, the suggested rules for pruning false positive detections are unrelated to the used pedestrian detector. This allows other researchers to use their preferred object detector and subsequently use a limited manual input effort in their application field to derive the correct post-processing rules.

3.4 Conclusion: benefits of using scene constraints

Chapter 3 clearly illustrates how we can use scene- and application-specific knowledge to derive pre- or post-processing filters, designed for improving the output of existing object detection algorithms. Hereby we mainly focus on using detectors with a sloppy probability threshold, allowing to have a high recall rate but ensuring as much objects as possible being detected, at limited precision meaning we know that the detector keeps triggering false positive detections. Our filters are designed specifically for reducing these false positive detections, thus improving the overall average precision obtained by our algorithms.

In section 3.1 we discussed an efficient pipeline for both detecting and classifying orchids in an automated sorting set-up. By using a boosted cascade of weak classifiers in combination with a decision tree of binary SVM classifiers, we succeed in classifying orchids 100% successful at plant level. We achieved the pre-set goal of detecting enough orchid flowers per plant, without triggering false positive detections at plant level *(thus providing multiple views of the same orchid flowers)*, to be able to correctly classify each plant in one of our five pre-defined texture-based classes.

Section 3.2 illustrates perfectly how object detection models can be built with a very small amount of data if we only need that model to work in that specific set-up. While our model is able to detect actual walking aids with an average precision of only 45%, by combining that with smart post-processing filters we achieve a high success rate at classifying walking sequences, pointing out if a walking aid has been used or not. The obtained result clearly helps the industrial partner to classify whether a walking aid has been used or not, and that with a high classification accuracy per sequence of 92%. This improves the research based on top of our classification, taking into account the used walking aid when analysing the gait speed of the person under investigation.

Finally, section 3.3 shows how any powerful off-the-shelf object detector can be combined with scene- and application-specific filters and classifiers, to obtain high average precision rates. This allowed us to build a robust pedestrian detection pipeline to allow automated blurring of pedestrians in mobile mapping images, safeguarding privacy as much as possible. However, this case immediately highlights a possible danger of using scene constraints. If we set the constraints to strict, we could loose possible detections of objects due to their strictness. E.g. if we set a constraint on the detection size related to the position in the image, but that relation is trained on adults only, we might not be able to detect children with the same interface, although they are also pedestrians. To cope with this case, one should always consider all possible conditions of the object class, eventually formulating multiple constraint-based

rules that cope under these different conditions. For the industrial partner using this automated privacy blurring solution, the provided solution helps drastically reducing the amount of manual labour. However, since we cannot guarantee an average precision of 100% for all types of pedestrians *(e.g. children or people in a wheelchair)* and bicyclists, due to the limitations of the used detector, this system cannot completely remove the need of manual intervention in privacy safeguarding, which would be the holy grail for the integrator.

In the following chapter, we will go one step further, integrating scene knowledge into our existing object detection pipeline based on a boosted cascade of weak classifiers, in order to mimic the behaviour of more advanced techniques like the Integral Channel Features detector of Dollár et al. [26]. We will prove that we can get equal or better results on several industrial applications by smartly combining all constraints into the object model training pipeline.

Chapter 4

Enhancing simple detection algorithms by integrating scene knowledge

The work presented in this chapter was published at the IPTA 2016 conference [93] and the GEOBIA 2016 conference [91].

In chapter 3 we introduced the concept of scene- and application-specific constraints. Furthermore we proposed several techniques that used these constraints as the basis for efficient pre- and post-filtering techniques. As discussed in section 2.1 several techniques tried integrating extra knowledge *(e.g. colour)* into the actual training process. Therefore we investigate in this chapter how we can integrate our scene- and application-specific knowledge into the actual training process. For achieving this task, we create our own feature channel(s), by transferring expert knowledge from scene constraints, onto the actual training data. This data can then, in turn, be used with our existing back-end, directly forcing the AdaBoost to select features that are related to our scene- and application-specific conditions.

The biggest difference between using ICF [26] as an algorithm directly and our proposed approach is as follows. ICF adds several feature channels to the training pipeline, basically telling the user *'the more the merrier'*. They use an adapted boosting algorithm that looks over all possible feature channels for the most discriminative features and selects those to form the actual detection model. However, it still requires to calculate a huge amount of possible features during training, from which many can be discarded by intuition *(e.g. looking*

for red coloured regions if we know the object's colour is actually green). By adding our scene constraints as processing steps on the training data, we help our AdaBoost system in the selection of strong and descriptive features, and explicitly tell the boosting where to look for them. This speeds up the training process and achieves similar accuracies as simply letting the AdaBoost system consider every possible feature in every possible feature channel. The only pre-requisite is that you know which kind of scene constraint will improve the training because adding specific filters can also remove valuable information that one might not have considered before. If you do not have scene-specific constraints available, then using the ICF algorithm directly for training would probably be a better approach.

We will investigate all these possibilities through three industrial relevant cases. In section 4.1 we will discuss our suggested integration approach for the application of robotic harvesting and harvest estimation and more specific to the cases of strawberry and apple picking. Following up on that, section 4.2 describes a similar approach for detecting photovoltaic installations in aerial footage. It continues on the integration approach and compares against several other techniques presented in the literature. Finally, section 4.3 formulates a conclusion on using this approach of integrating scene- and application-specific knowledge in the training process of object detection algorithms.

4.1 Automated visual fruit detection for harvest estimation and robotic harvesting

Autonomous robotic harvesting is a rising trend in agricultural applications, like the automated harvesting of fruit and vegetables. Farmers continuously look for solutions to upgrade their production, at reduced running costs and with less personnel. This is where harvesting robots come into play. While the mechanics of grabbing objects is a well-documented problem with many proposed solutions, robustly localizing objects stills seems to be very challenging, due to natural variations in shape and size, occlusion and uncontrolled lighting conditions. Multiple solutions, discussed in more detail in section 4.1.1, tried tackling the detection and localisation of single objects, yielding only moderate success.

Aside from investigating object detection as a localisation mechanism of fruit instances, we also improve the detection of single object instances within clusters, by suggesting two approaches for separating clusters into individual object instances. Of course, this complete pipeline is speeded-up using scene specific knowledge, which is the main theme of this dissertation.

4.1.1 Related work on visual fruit detection

The state-of-the-art in automated fruit localisation focusses either on 2D segmentation based approaches, like the work of Stajnko et al. [107], or adds extra information to the problem using either a thermal LWIR camera [108], a hyperspectral sensor [79] or even the power of 3D scanning techniques [123] to add an extra layer of knowledge to the object detection approach. The downside of adding extra layers of information is the increase in computational complexity and thus these techniques become quite time-consuming, especially when the approach includes calculating a full high-density 3D point cloud. Using hyper-spectral imaging also demands expensive hardware, with only limited resolution at reasonable prices. Therefore one of the main goals of this section is to prove that 2D information can be sufficient for robust object localisation. Many of these additive techniques are quite depending on very controlled (lab) environment settings, that are impossible to achieve in the open air or greenhouse agricultural set-ups, and thus they are not usable in our application.

The segmentation based approach, suggested by Stajnko et al. in [108] uses thermal imaging (LWIR) for the counting and analysis of apple cultivars in orchards. Their research is based on the different IR radiation patterns between fruit and leafs. To ensure a decent detection accuracy, images need to be acquired in the afternoon, to achieve a large temperature gradient between the apples and the background. This is a downside when creating a universally applicable approach since fruit exposed to less direct sunlight, results in partial and incomplete apple detections. Subsequently, a simple colour channel based segmentation approach is used, requiring the definition of hard thresholds, which could vary over time. This approach is further improved by Stajnko et al. [107], where the possibility of using pure RGB based colour and shape segmentation for apple detection and analysis is investigated. Wherever the basic segmentation does not work, specific image transformations on separate colour channels are applied to achieve a better contrast between leafs and fruit. Finally, they apply a parameter based blob analysis to identify fruit instances. This works fine for fully visible object instances but fails when objects get partially occluded and the resulting blob no longer has the correct parameters. In comparison with our approach, object categorization techniques are able to detect partially occluded instances, since these variations are also included in the training data, and thus ensure that more objects will be found.

Yang et al. [123] use a 3D stereo set-up, applying colour based segmentation on the intensity image, successfully separating clusters on tomato plants. We would like to retrieve individual objects, which is impossible for this cluster based segmentation approach, mainly due to the fact that separate instances in

a cluster share a similar depth profile.

Finally, Feng et al. [33] propose a fruit detection and classification method for a strawberry harvesting robot, closely related to one of our test cases. They introduce the use of OHTA colour spaces [78], on which they apply segmentation based algorithms to extract strawberries from the background, which is proved to work in single object instances. However, expanding this to a harvesting set-up on a real robot, is nearly impossible, because the segmentation will most likely not work due to object clusters, occlusions of leafs and different lighting conditions. Similar research in lab conditions is performed by Dubey et al. on individual object instances, against a clean background, successfully segmenting and analysing the apple region [28].

The downsides of segmentation based approaches can be solved by the proposed object detection algorithms. The work of [106] on recognising and counting peppers in cluttered greenhouses, based on a bag of visual words combined with a sliding window approach, is a first attempt of using more advanced computer vision techniques for solving the task. They start by locating fruit in individual images, then aggregate the estimates from multiple images using a novel statistical approach to cluster repeated and incomplete observations. Compared to our technique, where only a single view from the set-up is needed, this approach can only work successfully if multiple views are provided to support the different observation hypothesis.

4.1.2 Collected strawberry and apple datasets

Since no publicly available annotated datasets of fruit in unconstrained conditions exist, the first step in implementing our object detection algorithm is gathering the necessary data for training and testing the specific object models. We focus on two separate cases: strawberry picking and apple harvest estimation. Positive training data is gathered by collecting different representations of the strawberries and apples *(e.g. different illumination conditions, different orientations, partially occluded by branches and leafs, different viewpoints, ...)* whereas for the negative training data the objects are removed and the remaining image pixels are used as background information, maintaining application-specific background knowledge. This does limit the ability to use the trained models outside the intended set-up but it decreases the training time immensely compared to learning a set-up independent object model.

For the strawberry picking case, the goal is to localise all ripe strawberries given an RGB input image of the scene. A trinocular stereo set-up is used in order to grab different viewpoints *(bottom-up- and side-views)* of the strawberries. For the apple harvesting estimation application, two apples cultivars are tested:

Table 4.1: Data overview for both applications: number of images, number of annotations, model dimensions and the number of negative window samples.

		strawberry		**appleGala**		**appleRedDelicious**	
		train	***test***	***train***	***test***	***train***	***test***
pos	***#images***	205	750	30	30	32	32
	#labeled	1500	/	1595	625	1075	1160
	dimensions	35x38		65x65		62x66	
neg	***#images***	200	/	30	/	30	/
	#windows	5000	/	4000	/	3000	/

Gala and Red Delicious. For both cultivars, a training dataset is gathered using side-views of the apple trees inside the orchards. For evaluation of the apple models, images with a thirty degrees angle inclination were used, giving us a clear separate test set, maintaining the objectivity in evaluating the models.

Table 4.1 gives a detailed overview of the train and test images collected, the number of annotated objects, the model dimensions and the number of negative samples used at each boosted stage of weak classifiers. All annotations are manually made by a domain expert, to ensure the correctness between ripe and unripe strawberries. In the strawberry case, images were captured with two AVT Manta cameras both having a 1.292×964 pixel resolution. In the apple cultivar case, images were captured in the field by a third party using a Samsung NX3000 with a 3.648×5.472 pixel resolution.

4.1.3 Suggested strawberry and apple detection approach

In this subsection, we describe the complete pipeline for building both the strawberry and the apple detector models. We also prove that using scene constraints either during or after the training process increases the accuracy of the detection output. The approach itself is developed using the data of the strawberry case, whereas the data of the apple case is used as validation, to verify the approach for similar agricultural cases.

A boosted cascade of weak classifiers on original data

To start off we build a boosted cascade classifier using AdaBoost [120] with local binary patterns (LBP) [64] as local feature descriptor. Furthermore, each input sample, evaluated by the learned model during both training and validation phase, is pre-processed using histogram equalisation to account for varying lighting conditions. For each captured image from the trinocular stereo set-up, a set of manually positioned annotations is supplied, for which the LBP features

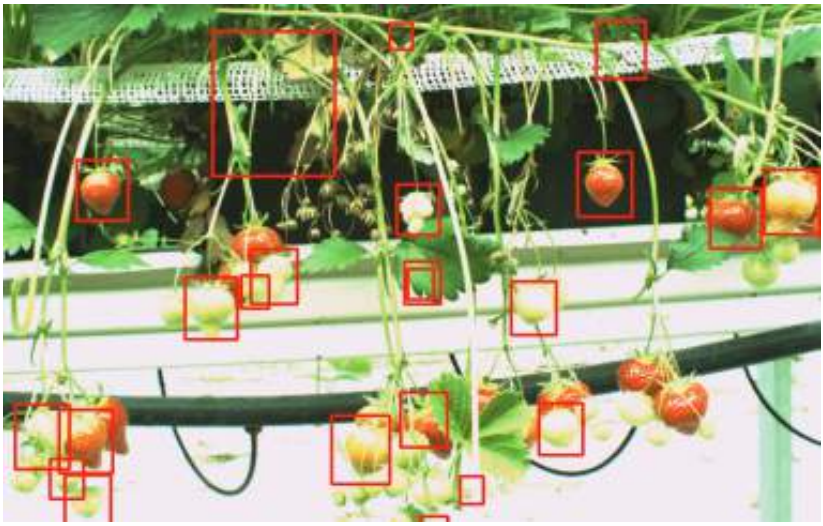


Figure 4.1: Strawberry model trained with both ripe and unripe strawberries.

are learned and used by the boosting process to optimally separate the positive and negative training samples in that given LBP feature space.

We train two models using the strawberry training data. We start with a model using all strawberries (both ripe and unripe) annotations. This results in a model, able to separate strawberries from the background and the plant with mediocre results, as shown in Figure 4.1.

Since our focus is to separate ripe from unripe strawberries, this model does not suffice, because it would still need a second colour based classification step. To cope with this we train a second model where the unripe strawberries are used as negative training samples. This approach does not yield decent results, since colour information is simply ignored by the proposed boosting process based on the LBP feature descriptor, and thus no feature based separation can be made between ripe and unripe strawberries.

We tested a similar approach on the apple case since ripeness was not an issue here. We noticed that training a model on LBP features yielded quite a lot of false positive detections while missing actual apples. This is mainly due to the simple shape and structure of an apple, yielding feature descriptors that are not unique enough compared to the background information. This problem can only be solved by adding more scene- or application-specific information, like for example the known colour of the ripe fruit.

Adding Scene Specific Information to Improve The Detector

To avoid valuable colour information being ignored and in the meantime trying to reduce false positive detections, we investigate possibilities to include colour information as an extra scene- and application-specific filter. This idea is inspired by the work of Dollár [26], and was already proven successful in chapter 3. The addition of colour information should make it possible to separate between dark red ripe and green unripe strawberries and to post-filter valid red apple detections, in order to drop the amount of false positive detections. To incorporate colour information into our set-up, we derive an application-specific colour transformation from the fact that both ripe and unripe strawberries have unique separable colours, as seen in equation 4.1. Furthermore, a hyperspectral study of the fruit showed that the best separation between ripe and unripe strawberries was obtained by comparing two spectral bands close to the red and green colour spectrum.

$$I_{RG} = \begin{cases} 0 & if \quad I_R - I_G < 0 \\ I_R - I_G & if \quad I_R - I_G > 0 \end{cases} \tag{4.1}$$

The derived equation supports red coloured regions, while it punishes the greener regions, by subtracting the green channel I_G from the red channel I_R and clipping negative values. This basically boils down to projecting the RGB colour cube on an axis connecting these two colours G(0,1,0) *(representing leafs, branches and unripe strawberries)* and R(1,0,0) *(representing ripe strawberries)*.

The general idea can be applied to any coloured object with a distinct colour difference to the background. Applying the equation results in an image with a visible difference between ripe and unripe strawberries on the one hand and background information on the other hand. This is visualised in Figure 4.2 and only applies if the background does not consist of bright reddish colours, similar to ripe strawberries *(greenhouse conditions)*.

Figure 4.3 illustrates the different ways we can integrate this extra colour based knowledge. The original model trained on RGB data, only looking at LBP features, can be seen in Figure 4.3(a). Using the information as a post-processing filter after applying a general colour ignorant object detector, is seen in Figure 4.3(b). This works in case of the strawberries, where building a cascade classifier for ripe strawberries only is not feasible, as shown in Figure 4.1. After each colourless detection, a post-processing filter can define if in the I_{RG} image the response is high enough to decide if the detection is an actual ripe strawberry. We apply an Otsu thresholding [80] on the I_{RG} image to receive a binary image and calculate the number of white pixels.

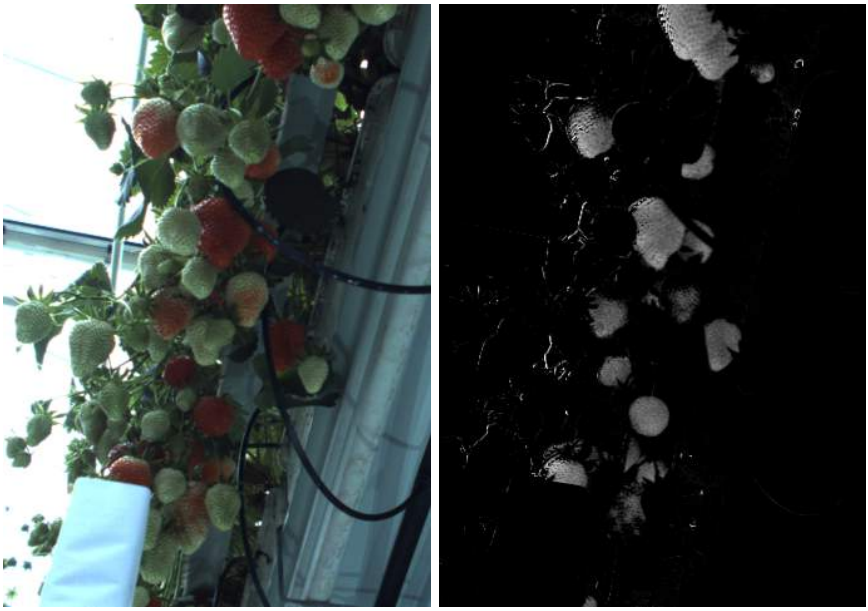


Figure 4.2: I_{RG} colour transformation applied to the RGB input data, which is used to segment ripe strawberry areas. *(normalized visualisation over full [0-255] pixel range)*

If more than 50% of the detected pixels yield a response in the binary map, we allow the detection as a ripe strawberry.

This post-detection candidate removal leads to an increase in processing time due to an extra filter step but reduces the amount of false positive detections. To avoid this increase in processing time, we can also apply the colour based knowledge as a pre-filtering of the training data, by retraining the object model on both I_{RG} transformed positive and negative datasets. Compared to using a post-processing filter this results in faster processing, a higher amount of true positive detections and a lower amount of false positive detections, as visualised in Figure 4.3(c).

Since our object detection model is applied on a multi-scale basis, we restrict the object size search range of the object detector, due to the known distance of the camera set-up compared to the object itself. This again results in a speed-up and a decrease in false positive detections, since multiple layers of the processed scale pyramid during multi-scale detection can be ignored in the search for object candidates. As seen in Figure 4.3, it only makes sense to detect apples in a certain range, depending on the distance to the camera, removing any false positives of undesired scales.

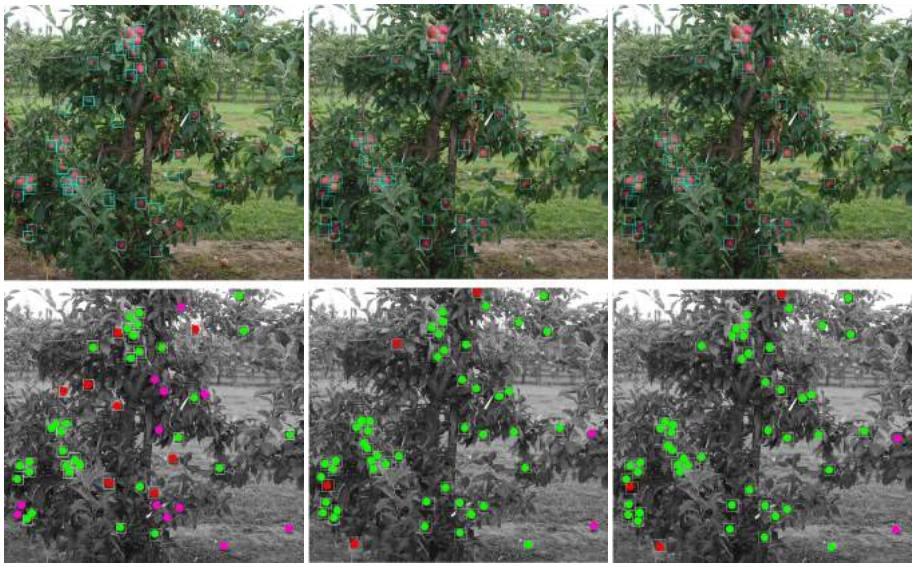
Splitting object clusters into separate object instances

A reoccurring issue during fruit localisation is the existence of fruit clusters. Segmentation based approaches are in most cases unable to find the small borders between connecting objects. This is where our object detection framework comes in handy. Since overlapping and partially occluded objects are also part of the positive training set, the model is able to identify single objects within clusters.

This means that the output of the detector can be used to successfully separate clusters. We suggest two possible approaches following our object detection pipeline, that use the location of the found detections to segment object clusters.

1. *Watershed-based segmentation*

Applying Otsu thresholding on top of the I_{RG} image results in a binary mask with possible ripe strawberry pixels. Next, the object detection returns regions of interest containing individual strawberry detections. White pixels in the



(a) RGB input data

(b) RGB input data + I_{RG} post-processing.

(c) I_{RG} pre-processed input data.

Figure 4.3: Adding scene knowledge either as separate processing filter or integrated into model training through training data preprocessing. *(top row)* actual detections *(bottom row)* detections shown as TP (green), FP (red), FN (magenta).

Figure 4.4: Watershed-based segmentation for separating object clusters.

mask, falling together within a single object detection bounding box, are first of all merged together. This copes with the negative effect of the colour transformation, where clutter covering parts of the object, splits single object instances into separate binary blobs. Once a merged binary image is produced, the centres of all detections are used as initialization positions for a watershed-based segmentation [11], combined with a separate randomly positioned seed point for the background. The watershed splits larger blobs into separate objects and identifies them with a unique ID, as illustrated in Figure 4.4. The borders of each region are defined by the merged binary image, pinpointing areas with possible strawberry pixels. A downside of the technique is the harsh boundaries between two objects, however, this depends on the implementation used.

2. Trinocular stereo triangulation based segmentation

To avoid the harsh boundaries of the watershed based segmentation we propose a second solution, based on a calibrated trinocular stereo set-up, like in our strawberry case. By performing the strawberry detector in all three camera views we know where strawberries are located in the given 2D images. We then apply a Difference of Gaussians (DoG) filter on the found detections (see Figure 4.5), which locates the strawberry seeds. By performing 3D triangulation on those seed locations, a local 3D map of the strawberries is generated. Combined with the segmentation data from the I_{RG} image, the depth edges can then be used to separate strawberries in clusters.

Evidently, this approach only works for objects that have identifiable unique textures, yielding enough unique points for the 3D triangulation. In case of flat texture-less objects, the watershed based segmentation approach will still achieve better results.

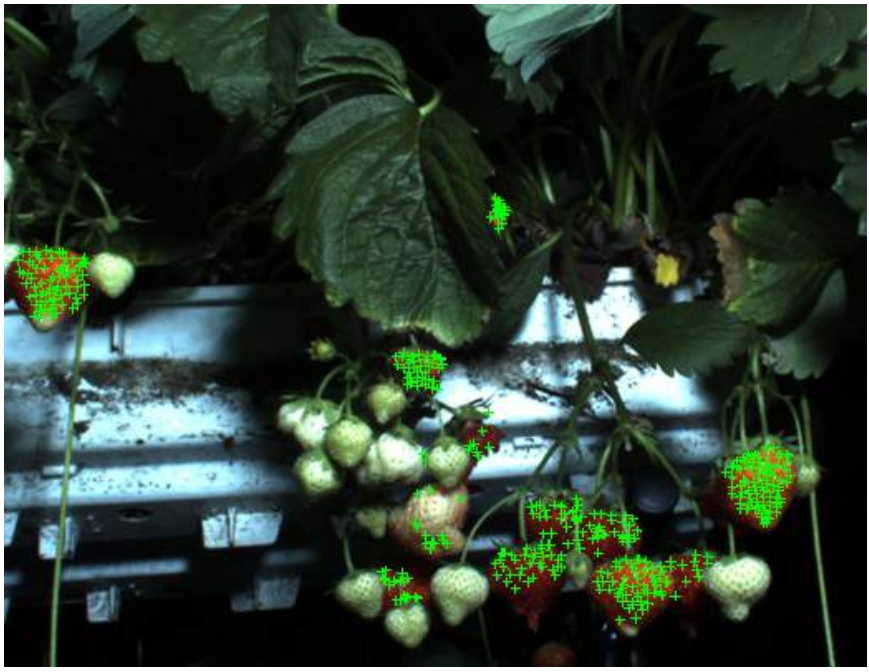


Figure 4.5: Applying DoG filtering to identify seed positions inside detections.

4.1.4 Results

Results on the strawberry test case

Due to the lack of decent ground truth data provided on the strawberry test sets, producing objective accuracy results on the produced object detectors is not that straightforward. One of the main challenges lies in defining what we have to classify as a ripe strawberry and what not. This differs from strawberry to strawberry cultivar, and thus introduces a bias when people start annotating ripe strawberries. Even application experts have troubles uniquely defining a ripe strawberry using objective criteria.

Visual results, as seen in Figure 4.6, clearly show that we are able to accurately detect visible and partially covered strawberries. Furthermore, all 'pick-able' strawberries are clearly located. Much depends on the extent of the used training set, and the quality of the annotation inside that given set of images. We clearly notice that using application-specific colour information improves the detection output and that we are able to uniquely identify objects inside object clusters, which can be used as seed points for our segmentation approaches.

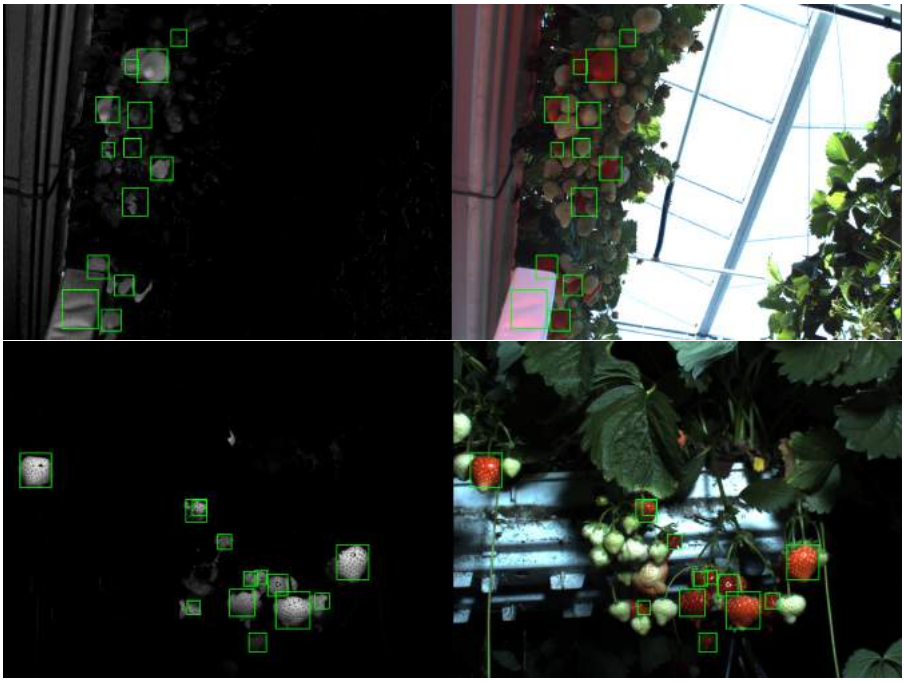


Figure 4.6: Visual detections for the strawberry picking case in both viewpoints.

Results on the apple cultivar test case

In contrast to the strawberry case, the apple cultivars test sets are accompanied by a complete set of manually provided object annotations. This allows us to perform a qualitative accuracy analysis in the form of precision-recall curves, as seen in Figure 4.7. For both apple cultivars, Gala and Red Delicious, we evaluated both grayscale *(striped curves)* and I_{RG} *(filled curves)* models on the available test data.

We clearly prove that adding the I_{RG} colour transformation to the model training pipeline improves the detection performance of the model compared to using the raw grayscale input image data, pushing the average precision higher. We compare this to a pure segmentation based approach, seen as the black curve inside Figure 4.7. For this we apply a hard threshold on the I_{RG} image, followed by erosion and dilation operators to remove noise and a blob detection algorithm. Each detected blob is evaluated on its dimensions, ensuring that only blobs inside a specific scale range are allowed. Visual results in Figure 4.8 clearly show a well-performing algorithm in challenging conditions like occlusion, background clutter, etc.

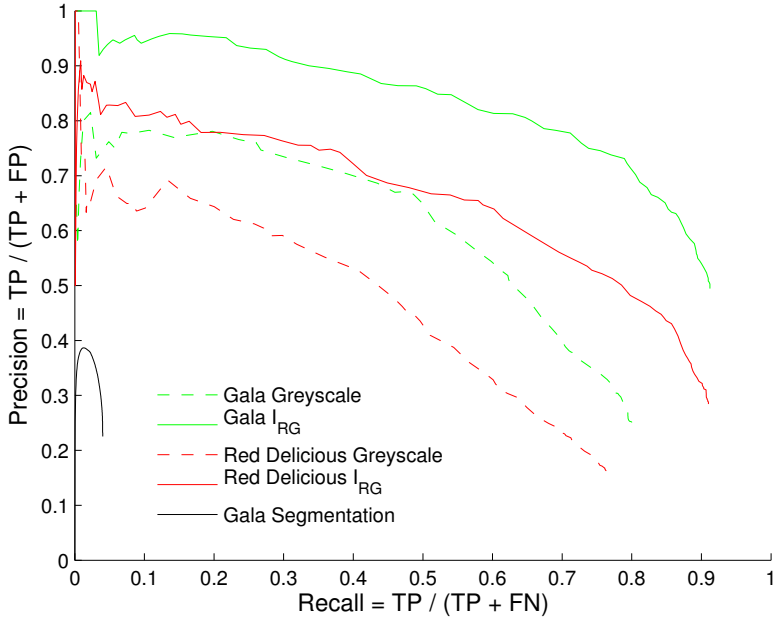


Figure 4.7: Precision - Recall curves for both apple cultivars.

4.1.5 Discussion and conclusion on the fruit detection cases

While existing segmentation based object detection approaches seem to work for agricultural applications in very constrained 'lab' conditions, they tend to fail when the environment gets more challenging. Adding extra sensors like IR, hyperspectral or 3D sensors could cope with this, but have proven to work less efficiently in outdoor conditions than 2D image processing. This was our motivation for developing a robust and promising algorithm for agricultural object detection. By smartly combining an object detection framework based on a boosted cascade of weak classifiers, with scene- and application-specific pre-filtering and efficient cluster segmentation, we created a promising pipeline for applications like strawberry picking, apple harvest estimation, ... Furthermore, the lack of publicly available and annotated datasets limits the possibility of comparing to existing techniques in the field.

Figure 4.8: Visual detection results for Gala (first row) and Red Delicious (second row) test images. All detections indicated with a bounding box and a filled detection circle.

We discover the problem of annotation bias, whereas the annotator and the person in charge of training the object detector need to agree on how the objects in the image should be annotated. If not, different shaped annotations, covering more or fewer object pixels, can lead to very misleading results on the precision-recall curves, showing worse results than the actual accuracy of the trained object detector. It also became clear that defining the actual objects to be found can be very challenging, even for domain experts.

4.2 Detection of photovoltaic installations in RGB aerial imaging

In this section, we discuss another case where we compare a boosted cascade combined with constraints approach against a channel-based approach. To prove the power of these object detection based approaches, we also compare against several more naive approaches like colour-based segmentation and maximally stable regions (MSER).

Solar panels provide a plausible solution for generating energy from non-polluting resources and therefore placing a solar panel installation is subsidized and encouraged by governments and electricity grid administrators. However, due to the given funding and tax reduction, these 'greener' energy-generating alternatives also give rise to malicious fraud. Our client, an electricity grid administrator wants to use the power of computer vision to track down these fraud cases to ensure that no financial benefits are given to people that are not correctly registered at the grid administrator with their solar panel installation.

Fully automated solar panel detection in RGB images yields major challenges, as seen in Figure 4.9. First of all, a solar panel is an object shape with only a few distinct visual features like shape and colour, due to its simple shape and colour distribution. Secondly the images that are freely available on the FGIA (Flanders Geographical Information Agency) portal have a very limited resolution *(only 25cm/pixel)* resulting in a solar panel size of only 9×7 pixels, which is very limited for training a detection model based on the boosted cascade principle and visible in Figure 4.9(a). Due to the material properties of solar panels, specular reflections tend to change the visual properties of the panels, depending on the position of the sun. This could lead to solar panels

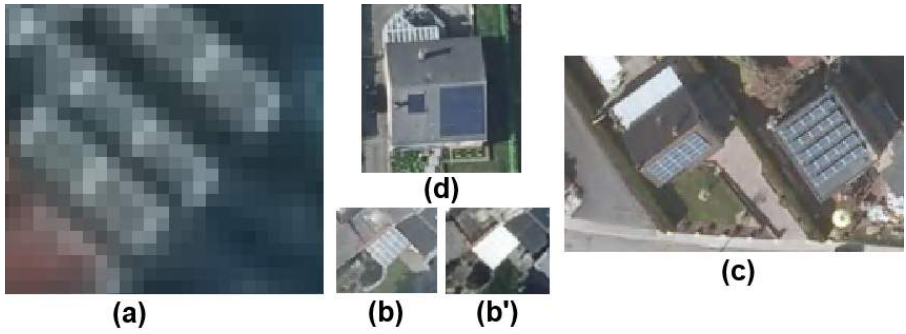


Figure 4.9: Challenges when dealing with solar panel detection.

occurring as a flat white overexposed part of the image. This difference can be seen in Figure 4.9(b) and 4.9(b′). The orientation in which the solar panels are placed also varies *(south, south-east, south-west)*, raising the need for a full 360° rotational search of the image for object instances. However, if one can ensure symmetry in the solar panels, a 180-degree range can be more than enough. Combined with the different roof slants (30°, 45°, 60°, etc.), panels get optically deformed and thus appear shorter or longer in 2D images, despite the fixed physical size, as seen in Figure 4.9(c). Finally, solar panels come in different materials *(mono- and polycrystalline, full black, etc.)* introducing, even more, intraclass variance into the problem. For example, due to the small resolution per panel, full black installations appear as a black square, yielding no visual features to train computer vision techniques, as seen in Figure 4.9(d).

Our goal is to develop a fully-automated computer vision based approach that is able to automatically detect solar panels in aerial imagery and return the location of these installations with a high probability. This, in turn, can avoid putting in huge amounts of manual labour to locate solar panel installations for fraud detection.

4.2.1 Related work on the detection of photovoltaic installations

The automated analysis of solar panel installations from aerial images using techniques of the computer vision field, limits itself to the analysis of solar panel efficiency and defects, using RGB and thermal cameras [100, 60, 116], while automated solar panel detection and localisation seem to be unexplored territory. While object detection is a well-studied problem in the field of computer vision, many other fields still have not discovered the power of these state-of-the-art techniques in autonomous object detection and localisation. The community of computer vision has however already performed object detection research in the field of aerial imagery, focussing on roads [44], buildings [72] or vehicles [39], always using state-of-the-art computer vision algorithms and thus giving perfect example cases that can be expanded to solar panel detection and localisation.

While the intuitive way would be to use segmentation based approaches, where the RGB input image is transformed to a colour space where separating object pixels from background pixels using strict (learned) thresholds is easier, they tend to fail when a wide range of other objects in the images have similar colour ranges. Furthermore, these techniques only take into account object colour information.

4.2.2 Datasets

For developing and testing our approaches, we use the freely available medium-scale aerial footage from FGIA (Flanders Geographical Information Agency) covering the grid area of Flanders where our electricity grid administrator is active. From this publicly available dataset, a set of 2.500 individual solar panels are manually annotated and a set of more than 150.000 random negative samples (containing everything except solar panels) are collected and used to learn the object detection models.

To test the four suggested approaches an aerial image of four square km of the city centre of Sint-Truiden is obtained. At a resolution of 25cm/pixel this results in a 8.000×8.000 pixel image, which is up-scaled using a bi-cubic operator to 16.000×16.000 pixels to ensure the solar panels are covered by enough pixels per panel. We reckon that this is mainly due to the software back-end used and not due to the fact boosting cannot learn a model based on fewer pixels per object. Inside this test image 313 solar panel installations are manually annotated, by drawing polygons around the installations, to use as ground truth when validating the fully automated object detection techniques suggested in this work. Due to the existing layout of solar panel installations, retrieving single solar panel annotations from these polygons is quite straightforward. The complete dataset, including training data, test data and annotations can be found for research purposes at `http://www.eavise.be/SolarPanelDataset/`.

4.2.3 Used approach

To find an optimal solution for fully automated solar panel detection in aerial images, a comparative study was performed. In the following subsection, each of the four state-of-the-art approaches is discussed in detail. Subsection 4.2.4 takes a closer look at the accuracy and time complexity of each individual technique.

Pixel-based colour classification using support vector machines

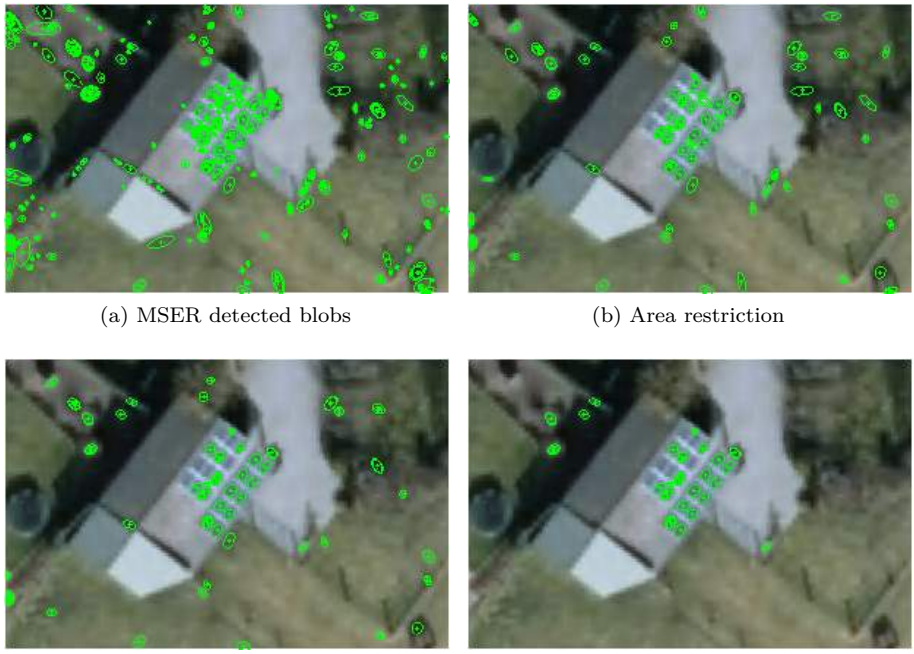
Our pixel-based colour classification, as seen in Figure 4.10, uses the internal coloured area of each solar panel *(blue-grey colour range)* without looking at the bright edges of the panel, in order to ensure that the pixel colour distribution of the training pixels is separable in the HSV colour space. We manually collect 1.000 internal solar panel pixels and 2.000 randomly selected non-solar panel background pixels. Both solar panel and non-solar panel pixels are transformed to the HSV colour space, where separating colours is easier than inside the RGB colour space.

Figure 4.10: Example of pixel-based colour classification using shallow learning as the learning tool. (left) original image (middle) pixel classification result (right) cleaned up segmentation.

Based on this training data, a support vector machine classifier with a linear kernel is trained, able to autonomously separate solar panel from non-solar panel pixels. When a test image is presented to the support vector machine classifier, each pixel is processed and is given a probability score, indicating how certain we are that the pixel is actually part of a solar panel installation. This probability score allows us to set a minimal certainty threshold, generating a binary image as seen in Figure 4.10. On top of that, binary image opening and closing operators are applied to remove noise, followed by contour detection and contour filling to achieve a cleaner result.

MSER based colour segmentation and shape analysis

Maximally stable extremal regions (MSER) [69] is a technique used to detect blobs inside any given image. By systematically increasing the threshold on a given grayscale input image, from very sloppy to very strict, we create a set of sequential binary images. Inside those images, the algorithm looks for regions that stay stable over the different thresholds and then approximates



(a) MSER detected blobs

(b) Area restriction

(c) Ratio restriction

(d) Colour restriction

Figure 4.11: MSER based colour segmentation.

those regions by their fitted ellipse. Applied in the case of solar panels, this generates a selection of candidate blobs, as seen in Figure 4.11(a).

Due to the higher response of solar panels in the blue channel compared to the red and green channels, we only process the blue channel data, removing the need to explicitly convert into a greyscale image. However, we could directly plug in the previous technique, and use the range of auto-learned pixel values, to define which pixel range to take into account, rather than selecting a single colour channel. To further filter candidate regions, we start by discarding blobs of an incorrect size *(see Figure 4.11(b))*, then look for blobs with an axis ratio that deviates maximally 30% of the ideal 1.5 : 1 ratio *(see Figure 4.11(c))*. Finally, we apply HSV colour based segmentation on the remaining blobs, using the previously discussed technique.

The known size range of the blobs can be explained by the fact that aerial imagery is taken on a constant height, while the limited ratio deviation is explained by the fixed physical size of the solar panels. Finally since our

training set contains solar panels with a known and shared colour range in the HSV colour space, we can allow colour based segmentation for further filtering.

Boosted cascade of weak classifiers

The previous techniques require a very limited training time since most processing is done on the fly when providing new test samples. This is different for object detection techniques, where a model is learned from a set of positive object samples and a large set of negative random background samples. From each training sample, a set of specific features is learned that are smartly combined into a model, able to separate objects from non-object patches.

Our first object detection approach is built upon the framework of Viola and Jones [120], using a boosted cascade of weak classifiers *(simple decision trees)*. This technique is originally developed for efficient face detection, but recent advances in computer vision [85, 89] prove that this technique still achieves top-notch results in other application fields, focussing on more general object detection test cases.

Since solar panels have a colour range that has a higher response in the blue channel, we decided not to use a grayscale image, but the blue channel instead, explicitly forcing the framework to use colour information and thus forcing colour properties into the training of the object model. The framework trains a model for a fixed orientation, so we explicitly rotated all solar panel examples to a horizontal position, resulting in a model able to detect horizontal solar panels

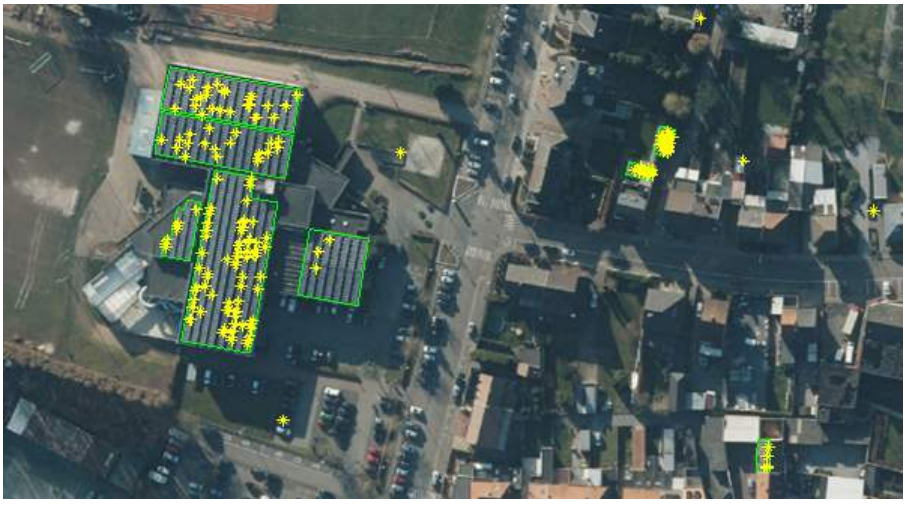


Figure 4.12: Detection of solar panels using a boosted cascade of weak classifiers.

only. However solar panels occur in different orientations, so we rotate the input image over different angles, with a predefined angle step, and then warp back the retrieved detections. This allows us to build a full 360-degree capable solar panel detector using a single orientation model. Running the detector on a test image, the detector will apply a fully rotational sliding window based evaluation of the image, triggering a detection at each position that gets classified as an object by the trained model. An example of such a detection output can be seen in Figure 4.12.

Aggregate channel features

The 'Aggregate Channel Features' - algorithm suggested by Dollár et al. [24] is, in fact, an extension to the Viola and Jones technique, incorporating colour information effectively in the training process instead of discarding it. This is one of the main reasons why we decided to test this framework for our solar panel detection case. Besides that, we also have an internally developed C++ implementation of this framework available, introduced in [18].

Running this more recent object detector on top of a given test image generates similar output as the previous technique, as seen in Figure 4.13. However, keep in mind that both Figure 4.12 and 4.13 are a sample detection output at specific detection thresholds of the algorithms. Deciding which algorithm performs better is done thoroughly in subsection 4.2.4.

Figure 4.13: Detection of solar panels using the ACF algorithm.

4.2.4 Results on the detection of photovoltaic installations

We started out with comparing our four state-of-the-art algorithms in processing time, as seen in Table 4.2. We see a clear difference in training time between the more basic HSV pixel segmentation and the MSER approaches, and the object detection approaches. While the boosted cascade and the aggregate channel features approaches take quite a bit longer to train on the given dataset, this task should only be done once because a trained model can be reused as many times as we desire. However, when looking at detection time, we notice a steady increase in processing time when computational complexity of the algorithm increases. Where the standard pixel based segmentation takes only 10 seconds for a 16.000×16.000 pixel image, the basic object categorization framework already takes 600 times that long.

These timings should be interpreted with caution because they are highly depend on the available infrastructure, which is also specified in Table 4.2 and the input resolution of the images. Furthermore, the implementation of the aggregate channel features technique is developed in-house and is not yet optimized for parallel processing, and thus processes everything in a sequential order.

One of the main reasons why object detection techniques take a lot more time, is that they are trained for a specific orientation. In order to be able to detect objects in every possible orientation, we rotate the original image for a full 360-degrees, with a single degree step. However, if computational efficiency is the end goal, we can reduce the amount of angles on which we evaluate the solar panel detector. We know a single model is robust to slight variations of about 20-degrees. This can already reduce the amount of steps to eighteen. On top of that we can argue that if we know where the north is given our image, we can exclude several orientations, since they will never be used for solar panel installations. This can probably again reduce the amount of evaluation by half, to about 9 different angles that need to be tested. If we can then integrate symmetry, we can even further reduce the amount of verified orientations.

Table 4.2: Comparison of training and detection times combined with the complete system configurations used.

Algorithm	**Training**	**Detection**	**Cconfiguration**
HSV + SVM	10 sec	10 sec	Intel Core CPU i7-4500U
MSER	0 sec	100 sec	Intel Core CPU i7-4500U
Boosted Cascade	3.5 hour	10 min	Intel Xeon CPU E5-2630
ACF	36 min	6 hour	Intel Xeon CPU E5-2630

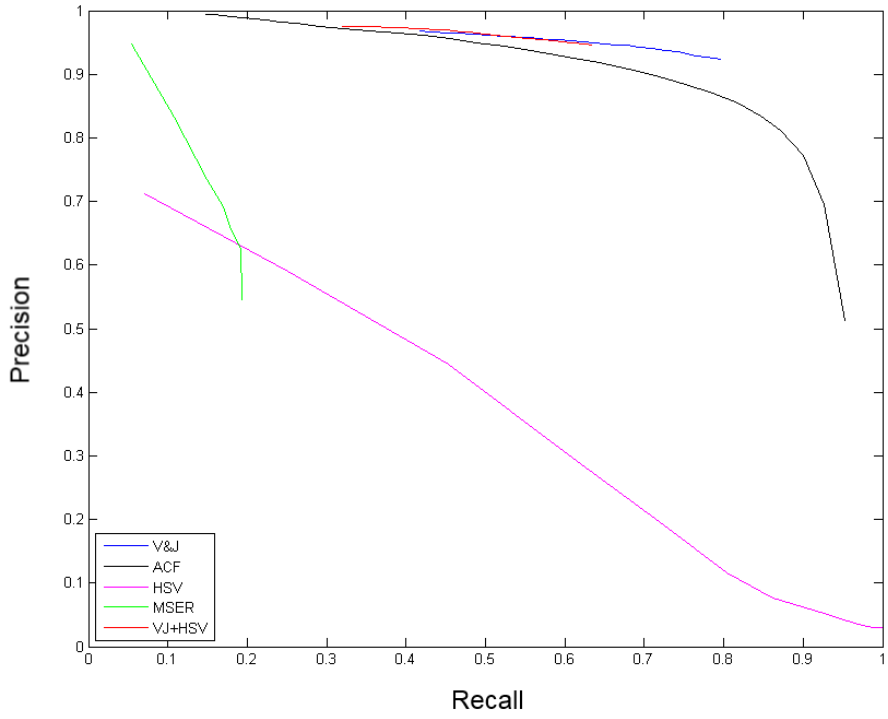


Figure 4.14: Precision - Recall curves for all techniques tested and validated on the 16.000×16.000 test image of the Sint-Truiden city centre.

To evaluate the accuracy of the implemented algorithms, we suggest to use precision-recall curves, used to compare the actual detection output with the manually obtained ground truth polygons. To generate these curves, seen in Figure 4.14, the generated detection maps are first downscaled to a 4.000×4.000 pixels, combining scores of the detections obtained on the same locations. On top of the resulting score map, a threshold is applied *(which is the varying parameter used to generate the different precision-recall values)*. This is followed by a dilation *(to make detection centres as large as solar panel dimensions)* and erosion *(to remove detections that are lonely and not grouped)* operations resulting in a clear binary image. This binary image is then analysed to calculate the true positives (TP), false positives (FP) and false negatives (FN).

Figure 4.15 shows the comparison of the binary output which is in turn compared to the manual annotations. By using the three channels of an RGB image, knowledge of both results can be visually combined.

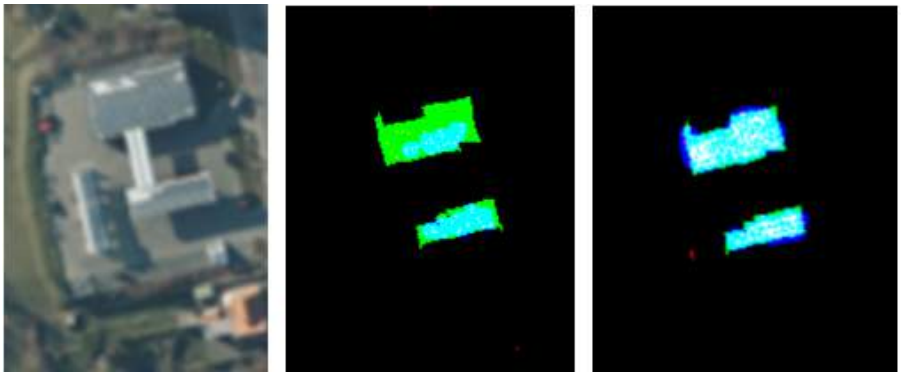


Figure 4.15: Score processing map for both boosted cascade and aggregate channel features algorithm. (left) original input image (middle) boosted cascade result (right) aggregate channel features result. *(pixel classification based colouring: cyan = TP / black = TN / blue = FP / green = FN)*

Detection centres are visualised as dots on the red channel, the ground truth is visualized on the green channel and finally the detection output regions are visualised on the blue channel. Combining those three colour channels yields a set of pixel based classification labels. Cyan labels indicate true positive detections, black labels indicate true negative detections, blue labels indicate false positive detections and green labels indicate false negative detections. The visualisation is done for both the boosted cascade and the aggregate channel technique.

The precision-recall curves clearly show that object categorization techniques outperform the other, more basic computer vision techniques. We notice that the boosted cascade technique performs still a bit better than the aggregate channel technique, which indicates that adding extra feature channels like colour and other gradient filters is overkill for solar panel detection. To test this theory we added HSV pixel based segmentation as a post-processing step to our boosted cascade detector, displayed as the red curve in Figure 4.14. We see almost no increase in efficiency which made us to decide to drop this extra processing, obtaining a smaller computational complexity and thus a faster running time.

4.2.5 Discussion and conclusion on the detection of photovoltaic installations

Our goal is to compare techniques for automatically detecting photovoltaic solar panel installations in RGB aerial images, considering very challenging conditions due to the limited resolution, a limited amount of visual features, specular reflections and different orientations of the solar panel object class.

Using pixel-based segmentation for solar panel detection yields only moderate results. In any given test image, it is impossible to only separate blobs that are pure solar panels, since the colour distribution of the panels, also tends to return in many background patches. Furthermore, there are several solar panel types that tend to deviate from the average colour distribution, like the full black type, generating even more missed detections.

Applying smarter feature-based techniques, like the MSER based approach, seems promising and even yields decent results in specific areas, but still has quite a fast drop in accuracy when considering larger regions where solar panels need to be detected, as seen from the resulting precision-recall curve based on the 2×2 km area of Sint-Truiden. However, our tests clearly prove that object categorization techniques like boosted cascades and aggregate channel features can outperform more basic techniques with the only downside that training data needs to be collected and that the single training set-up for building the model takes a bit longer.

We prove that computer vision is a working solution for fully automated solar panel detection in RGB based aerial imagery for fraud detection. For a fully automated system, we achieve a precision of 93% at a recall of 80%. Due to the heavy deformations to the solar panel object caused by the slant angle of the roof, training multiple models to cope with the missed detections could be the first improvement. Furthermore, we perform the research on mid-resolution images, at 25cm/pixel. However, there are already recordings of the same area at a resolution of 8cm/pixel, which would again yield an accuracy increase. Since our training set consists mainly of industrial solar panel arrays, it is also quite understandable that the model is not able to cope with the larger deformations found on solar panels placed on domestic housing. Improving the training dataset offers solutions here.

Finally, fully automated system should not be able to detect 100% of the solar panels with 100% precision. In most set-ups, an operator is still available. Lower probability yielding detections can be sent to the operator and in case of doubt be solved by manual inspection. This semi-automated approach would still reduce the manual labour drastically and reduce the time needed to process larger datasets.

4.3 Conclusion: benefits of integrating scene constraints

In this chapter, we explained how scene- and application-specific knowledge and constraints can be integrated into the actual object detection model, by training models on actual pre-processed data. This allows us to force detection models to look for a specific property of an object, instead of letting the training algorithm decide by itself if the feature is valuable for detection or not. By doing so, we force more basic object detection algorithms to more advanced object detection algorithms.

To prove our point, we provide solutions for three industrially-relevant applications as study cases. The first and second case focuss on fruit detection and localisation in agricultural applications in subsection 4.1, more specifically providing object detection models for strawberries and apple cultivars. This enables to create an automated harvesting robot, or harvest estimation system. We prove that our boosted cascade of weak classifiers with integrated scene specific knowledge in the training data can obtain highly accurate detection results. As shown in chapter 9 these fruit-based detectors are industrially used in the case of robotic strawberry picking, where the robot replaces human pickers whenever no-one is around in the greenhouse *(e.g. during night conditions or weekends)*. The combination of both robotic and human pickers leads to a faster collection of ripe strawberries without damaging the product. The robotic picker is mostly used for picking the clearly visible strawberries, whereas human pickers are used whenever the robot is failing, since we could not guarantee a 100% average precision, meaning that every single strawberry is detected.

Subsection 4.2 implements the third industrially-relevant application, where a similar approach for automated solar panel detection is used with the goal of obtaining an automated fraud detection system. Here we prove once again that introducing extra channels *(e.g. when using the ACF approach)*, is not always the best way to go. We rather suggest considering scene-specific knowledge to improve more stable and reliable detectors *(e.g. using the adapted Viola&Jones approach)*. While this system is not yet actively used by the electricity grid administrator who tasked us with the case itself, other parties showed interest into using our technique for a European project, focussing on robustly locating solar panel installations in aerial photography.

In chapter 5 we set the scene- and application-specific constraints part aside and look for possible solutions to reduce to amount of manual labour needed when training these powerful object detection algorithms. In doing so we investigate the possibility of using a technique called active learning and illustrate its use in efficiently training object detectors with minimal manual input.

Chapter 5

Integrating active learning to improve industrial object detection

The work presented in this chapter was published at the VISAPP 2017 conference [87].

In the previous chapters, we focussed mainly on integrating scene- and application-specific constraints into existing object detection pipelines. Although this already adds a lot of possibilities in training effective object detection models for industrial applications, scene- and application-specific constraints will never solve all the issues with object detection on their own.

One of the issues that still remains is that in many academically developed applications, high average precisions are reached by providing the learning systems with huge amounts of positive and negative training data. All things considered, this is not a big issue when the application is a class that is represented a lot in the existing large public datasets. In that case, it is simply a job of collecting all the relative images and their ground truth annotations.

However if your object class is different from the many that have off-the-shelf available models, which is the case in about every real specific industrial application, then basically the task of collecting and annotating training data has to be included in the design process. Collecting multiple thousands, even up to millions, of training samples and letting a user manually annotate all those images, is a time consuming and not to mention very expensive process.

This is one of the reasons why many small- and medium-sized industrial enterprises refrain from learning classification models. Luckily the academic community is aware of this problem and introduced the concept of active learning as a possible solution to the problem. The idea is that, when training a machine-learned classifier, the only training samples that really matter are the ones that shape the separation plane in the given feature space. So instead of simply providing millions of images, hoping that the crucial examples are included by providing numbers, active learning tries to actively look for training samples that actually matter. *(for a more detailed discussion on the active learning technique itself we refer to section 2.3)*

The remainder of this chapter is organized as follows. Section 5.1 applies the concept of active learning using our boosted cascade of weak classifiers back-end in order to generate improved open source face detection models. This is followed by section 5.2.2 discussing how we can apply this approach to our previous cases in order to obtain better industrial object detectors. It also highlights many of the benefits of active learning, in relation to these scene- and application-specific conditions. Finally, section 5.3 concludes this chapter.

5.1 Improving open source face detection by combining an adapted cascade classification pipeline and active learning

Face detection *(see Figure 5.1)* is a well-studied problem in computer vision, and good solutions are presented in the literature. However, we notice that open source computer vision frameworks like OpenCV [12], offer face detectors based on existing learning techniques, which are yet unable to yield equally high accuracies on the available public datasets as those reported in academic publications. The main cause for this can be the fact that most of these models have been created in the earlier ages of computer vision, when academic research was still interested in older cascade classifier based techniques, like the proposed algorithm of Viola & Jones [120].

Academic research evolved and moved on, discovering more promising techniques and losing interest in well established and proven-to-work algorithms. In the case of OpenCV, this resulted in a well-known computer vision library still providing quite a basic face detector, achieving only average detection results on any given dataset, without more state-of-the-art face detectors.

Figure 5.1: Example of *CascadeClassifier.detectMultiScale()* in *OpenCV3.1* framework (*OpenCVBaseline* model).

On the other side, users from the industry interested in turning these open source computer vision frameworks into working applications, get stuck at improving the existing performance of the face detection algorithms, because it seems the models already reach their maximal potential. Furthermore, the internal organizational structure of many companies does not allow to put efforts into research that tries to boost the performance of current algorithms. Two of the main issues when trying to improve these existing detection models are the availability of large amounts of training data and the achievable accuracy limitation reported by academic research, using this basic detection model.

In order to fill the gap, we decided to investigate how the on Viola & Jones based cascade classification pipeline for training a face detector inside OpenCV could be adapted to achieve a higher detection accuracy. We do this by:

1. Using an active learning strategy to iteratively add hard positive *(positive windows classified as negatives in the previous iteration)* and hard negative *(negative windows classified as positives in the previous iteration)* samples to the object detector training process. This is immediately our most influential change to the training pipeline and also the focus of this chapter.

2. Improve the gathering of training sample collection, to make it more intuitive and remove the overburden of samples of which we know they do not add any more knowledge to the detector.

3. Adjusting the face annotations to focus on the inner-face rather than focussing on the outer-face and thus removing undesired information.

We experienced that industrial applications of face detection tend to fail due to false positive detections, as seen in Figure 5.1, because post-detection processing steps depend on an actual face being available. In the case of a face recognition application, the face detection can be the basis for gathering training and test annotations [58, 121]. Therefore we aim at improving the available face detection model of OpenCV3.1, based on AdaBoost [35] and local binary patterns [64], aiming for a very high precision at an acceptable recall.

The remainder of this section is structured as follows. Subsection 5.1.1 presents related research, while subsection 5.1.2 discusses the used framework and datasets. This is followed by subsection 5.1.3 discussing the proposed approach in detail. Finally, subsection 5.1.4 elaborates on the obtained results while subsection 5.1.5 sums up conclusions.

5.1.1 Why wanting to improve old OpenCV functions?

The OpenCV framework is an open source computer vision framework providing a collection of techniques ranging from basic image segmentation to complex 3D model generation. It steadily grows in size by contributions from a community of both academic researchers and industrial partners, adding recent advances in the computer vision community, while trying to maintain the quality of the existing back-end. We notice that once the new functionality is integrated for a longer period of time and heavily used by the community, investments in improving the functionality tends to stop. This could be explained by the fact that the computer vision community has no interest in actual relevant industrial implementations, but rather in pushing the state-of-the-art even further.

Recent advances in computer vision solve face detection by using complex techniques like multi-task cascaded convolutional neural networks [125], convolutional neural networks combined with 3D information [63] or recurrent convolutional neural networks [51]. These techniques yield very promising results but tend to be fairly complex to implement in actual applications. There is still a lack of well documented and supported open-source software libraries that are easy to use.

Furthermore, we noticed OpenCV is paving the way of integrating these newer techniques, but up till now, their performance inside the OpenCV framework is still not as bug and error-free as desired by industrial companies.

The work of Viola and Jones [120] on face detection using a boosted cascade of weak classifiers has been around for quite some time. It is the standard frontal face detector for many industrial applications so far, like e.g. digital photo cameras. A downside is that many companies use the available software to train

Table 5.1: Training data overview for trained models.

Model	**#pos**	**#neg**	**#stages**	**#stumps**
OpenCVBaseline	xxx	xxx	20	139
BoostedBaseline	1.000	750k	26	137
IterativeHardPositives	1.250	750k	19	146
IterativeHardPositives+	1.500	750k	19	149

their own more complex face detection models, without sharing the models back with the community. This is mainly due to the fact that OpenCV operates under a BSD license, allowing companies to use the code without sharing back any critical adaptations or changes. With our work, we aim at improving the currently available frontal face model based on local binary patterns *(used as a baseline in this publication)* and achieve a model that is able to accurately detect frontal faces in a large variety of set-ups.

One could argue that working on such an old technique is basically a waste of time invested. However, several recent research papers like [126, 34, 103] prove the importance of such well-established techniques for specific cases of industrial object detection.

5.1.2 Framework and dataset

For building our approach we depend on the OpenCV3.1 framework, provided and maintained by Intel. We focus on using the *CascadeClassifier* object detection functionality in the C++ interface together with the *opencv_traincascade* application, containing all functionality for building a boosted cascade of weak classifiers using the approach of Viola & Jones [120].

Since the training data of the current OpenCV face detection models are no longer available, we collected a set of face images for training our own frontal face detection model. The images are collected from various sources like YouTube videos and by using a bulk image grabber on social media, image boards and google image search results. Remark that all of these images are unaccompanied by ground truth face labels. On top of that, we created a multi-threaded tool that can use an existing face detection model to efficiently search for valuable face data *(hard positive and hard negative data samples)* in any given video, by running the detector over the video, retrieving all detections and letting an annotator decide whether they are faces or not.

For training our new models, we manually annotate 1.000 face regions as positive training windows and combine this with 750.000 negative training windows, automatically grabbed from large resolution negative images not containing faces. As shown in Table 5.1, we increase the positives dataset for each new iteration with 250 extra hard positive samples, gathered from a large set of positive images, in which we know faces occur *(by ensuring we only collect video data with faces in all kind of conditions)*, using our active learning strategy. Whenever the initial detector is not able to find a face region, a manual intervention is required, asking for a face label, and adding it as a hard training sample for the following training iteration.

For validating our newly trained models and comparing them to the existing OpenCV baseline, we use the Face Detection Data Set and Benchmark (FDDB) dataset [49]. This dataset contains 5.171 face annotations in 2.845 images collected from the larger Faces in the Wild dataset [9]. The dataset focuses on pushing the limits of unconstrained face detection. In order to be able to obtain a decent baseline, we converted the existing image annotations into the OpenCV used format and made them publicly available at `http://eavise.be/OpenSourceFaceDetection/`. Since we have this complete dataset available and since we can compare to an OpenCV baseline detector, this dataset is ideal for testing our active learning strategy.

5.1.3 Used approach

In this subsection, we will discuss the different adaptations made to the existing cascade classifier training pipeline, leading to an overall increase in performance, as discussed in section 5.1.4.

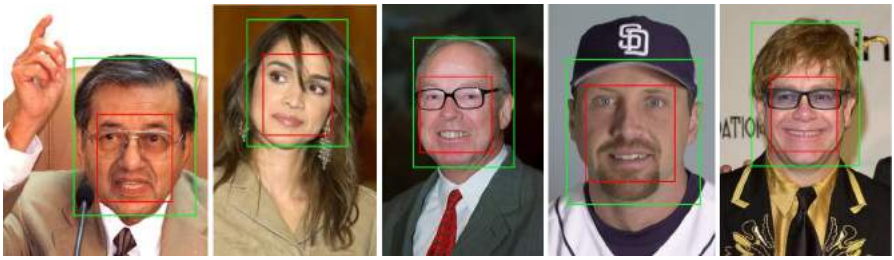


Figure 5.2: Changing the annotations from full-face to inner-face: (green) OpenCV (red) ours.

Changing the Face Region of Interest During Annotation

When taking a closer look at the output of the OpenCV LBP frontal face detector, we notice that in many cases the detection output contains the complete head, including ears, hair and sometimes even background information. This is due to the OpenCV training data annotations. Figure 5.2 indicates that OpenCV aimed to include as much facial information as possible to feed to the training algorithm. Since a face detector needs to be generic, we focus on the face part containing the most general features over any given face dataset. In order to reduce the amount of non-trivial face information, we decided to annotate faces as the inner face region only, seen as the red annotations in Figure 5.2, and as previously suggested by Mathias et al. [70] for similar face detection techniques. This approach has several benefits. It removes tons of features from the feature pool of the boosting algorithm, reducing the number of features that need to be evaluated during model training. Furthermore, the inner face is more robust to a rotation *(both in-plane as out-of-plane)*. We elaborate more on these in-plane and out-of-plane rotations later in this subsection.

Adapting the negative training sample collection

OpenCV offers an automated way of collecting negative samples from a set of random background images not containing the object. The algorithm rescales the given negative images to different sizes and uses a sliding window based sequential collecting of negative windows, without any overlap between subsequential windows. Once the set of negative images is completely processed, the process is repeated by adding a pixel offset in each image, to obtain slightly different samples *(at pixel level)*. If a set is traversed multiple times, increasing the offset each time, this process equals applying a pixel shifting sliding window approach, as illustrated in Figure 5.3. While the basic idea of capturing slight differences in your data might be a good starting point, this approach generates a huge amount of negative samples which do not add extra meaningful knowledge to the process and can thus not be seen as unique samples.

Looking at the boosting process used to train the cascade classifier (by default AdaBoost [35]), we notice that each new negative window can only be allowed as a negative training sample for a new stage if the previous stages do not reject it. If there is only a slight pixel shift for different negatives, then this rejection phase will just evaluate a lot of windows, of which we already know that they will be rejected. Therefore we adapted the interface and removed the pixel offset procedure. By removing this procedure and having no overlap between subsequent negative windows, we introduce a possible loss of valuable information shared around the borders of subsequent samples. This lost information might contain critical knowledge for building a robust detector.

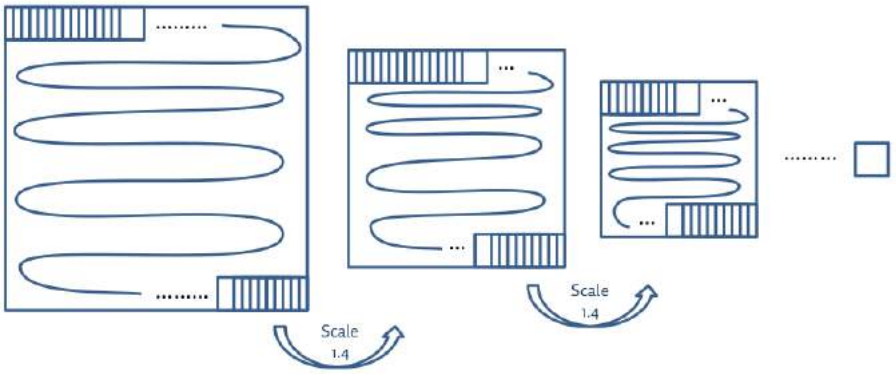


Figure 5.3: The original process for collecting negative training samples.

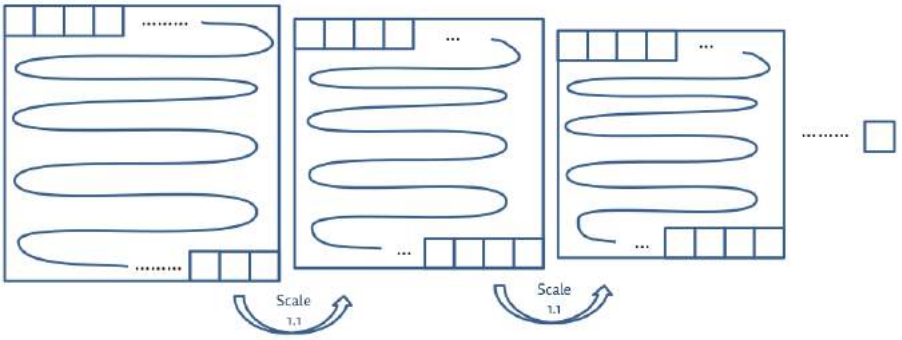


Figure 5.4: The adapted process for collecting negative training samples as suggested in this work.

To reduce this loss of information we refine the scale generation in the image pyramid. Where OpenCV generates an image pyramid with a scale parameter of 1.4, we decided to lower this scale parameter value 1.1 to ensure that negative samples gathered on different pyramid scales are diverse enough while keeping as much valuable information as possible. This is illustrated in Figure 5.4. By doing so, lost information on sample borders on one scale will be captured by either the previous or the subsequent scale. An extra benefit of refining the scale pyramid is that the resulting object detection model is more robust to scale changes of the object, able to capture smaller variations in size.

Based on these adaptations it is quite straightforward to collect a large set of negative data samples, something necessary to create a robust face detection model for in the wild applications. Considering a high-resolution image of 1.080×1.920 pixels, we can already collect 30.000 negative training samples. This allows us to increase the number of negative samples per stage in our trained cascade classifier to multiple hundred thousands of samples, trying to model the background as well as possible. This will increase training time per stage but will reduce the number of stages, and thus make the model faster and less complex. Since we also have more negative training samples available, we can select better features to separate training data and achieve a model that is more accurate at detection time.

Iterative active learning strategy for collecting quality samples

Supplying heaps of data to machine learning algorithms allows learning very complex object detection models. The downside is that both in gathering positive and negative training data, it is very difficult to tell which new sample will actually improve the efficiency of the detection model. In order to decide which samples are actually valuable to be added to the process, we apply a technique called active learning. The idea is to use the model trained by the previous iteration and use that model to tell us which samples are valuable *(close to the decision boundary)* and which are not *(no ambiguity in labelling)*, when adding them to the next iteration training process, as previously visualized in Chapter 2, Figure 2.9. We make a distinction between hard negatives and hard positives as explained below. Furthermore, the advantage of active learning is that we limit the amount of manual labour drastically since we only need to provide labels to new training samples that add extra knowledge to the trained classifier.

1. *Hard negative samples*

Hard negative samples are gathered by collecting a set of negative images and running our previously trained face detector on them. All detections returned are in fact negative windows that still trigger a detection, and are thus not assigned to the background yet by our current model. Basically, these samples contain information that was not yet captured by the previously collected set of negative samples and thus provide valuable information to the training process.

2. *Hard positive samples*

Hard positive samples are gathered by collecting a large set of unlabelled images containing faces. We only know the images contain one *(or more)* faces, but we do not have a labelled location. On these images, the current face detector is executed *(with a low detection certainty threshold)* and a piece of software keeps track of images that do not trigger a detection. In that case, an operator

is asked to manually select the face region for those triggered images and thus provide labels. This region is stored as a hard positive sample that can still give the model learning interface enough extra valuable knowledge on how it should be learning its model.

Halting training when negative dataset is consumed

The original OpenCV implementation uses pixel-wise offsets in the negative sample grabbing to avoid the training to halt when the originally provided dataset is completely consumed in a first run. We do not allow this and halt the training when the negative dataset is completely consumed. Once that happens we give the operator two possibilities. Either we allow to add extra images to the negative image dataset, or we return the number of negative samples that was grabbed in the last stage before the training was halted. This allows the operator to finalize the last stage with this exact amount of samples and thus train a model using every single negative sample window, completely consuming the available negative dataset.

Using the adaptations to train different face detection models

By smartly combining all these adaptations we train different face detection models where we iteratively try to improve the accuracy of the obtained model. Table 5.1 describes the training data used for these models, in combination with the number of model stages and the number of features *(each forming a stump/binary decision tree)* selected by the boosting process. One might argue that using more complex decision trees is more profitable but previous research shows that using more complex trees actually slows the detection process [99], because more features need to be evaluated in early stages.

Our first model *(referred to as 'BoostedBaseline')* can be seen as the baseline we iteratively try to improve by applying the active learning strategy. For each boosted learning model, the increase in performance when adding features should outweigh the complexity and thus the processing time.

For the second model, referred to as *'IterativeHardPositives'*, we add 250 hard positive training samples collected through the active learning procedure, trying to improve the recall rate of the detector. We also gather a limited set of hard negatives and add those to the training set. We noticed that adding these extra quality samples pushes the recall further drastically while slightly increasing the precision. The third and final model, referred to as *'IterativeHardPositives+'*, is again improved by providing 250 extra hard positive samples, in an attempt to push the reported recall even further.

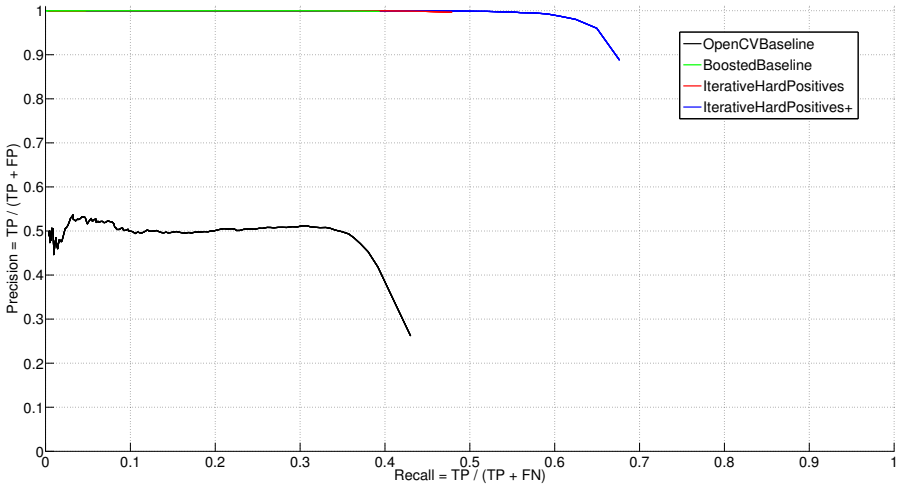


Figure 5.5: Precision-Recall for all models on FDDB dataset.

5.1.4 Results after active learning

Performance of trained models

Figure 5.5 compares the trained models (*BoostedBaseline*, *IterativeHardPositives* and *IterativeHardPositives+*) from section 5.1.3 to the *OpenCVBaseline* detector on the FDDB test dataset. Performance is measured using precision-recall plots. We notice a generally large improvement of our self-trained models *(green, red and blue curve)* over the OpenCV baseline *(black curve)*. The OpenCV baseline model is only able to achieve a recall of about 40% *(meaning 4 out of 10 objects are detected)* at a precision of 40% *(of all the detections returned, only 4 out of 10 are actual objects)* for its optimal point. Of course, one can make a trade-off and decide to sacrifice recall for a higher precision. Nonetheless, the current OpenCV model is not able to detect objects with a certainty over 50% on the given FDDB dataset, containing a wild variety of faces in challenging conditions.

Compared to the *OpenCVBaseline* detector, at the optimal recall of 40% for that model, our *BoostedBaseline* detector already increases the precision towards 99.5%, almost completely removing the existence of false positive detections. Furthermore, each of our subsequent models, as seen in the close-up in Figure 5.6, increases the recall further without sacrificing the very high precision rate. At a recall value of 60%, a 50% increase compared to the *OpenCVBaseline* detector, our *IterativeHardPositives+* detector only has a slight drop to 99% precision. As an optimal working point, our *IterativeHardPositives+* model reaches a precision of 90% at a recall of 68%.

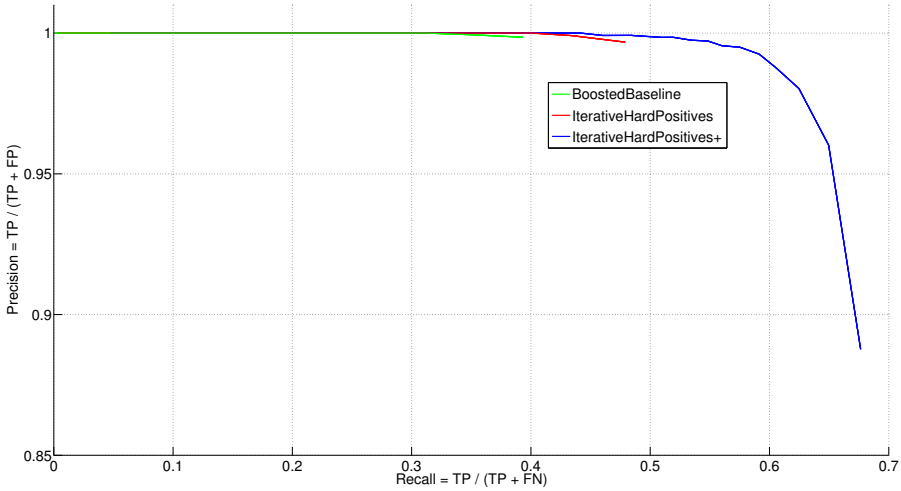


Figure 5.6: Close-up of PR curves of our detection models.

While many papers on face detection use precision-recall curves to compare detection models efficiently, the official FDDB evaluation criteria are based on the true positive rate compared to the number of false positive detections.

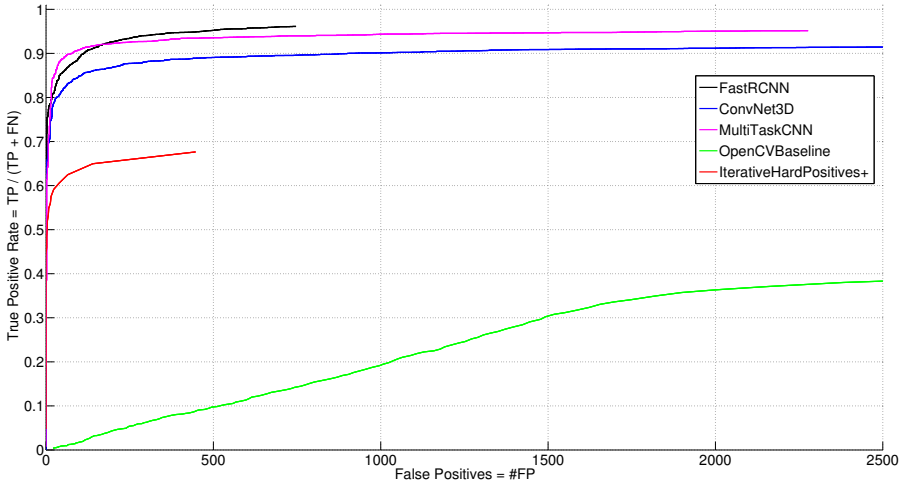


Figure 5.7: Evaluation on the FDDB dataset, comparing our algorithm to neural network based approaches.

We include this comparison for both the *OpenCVBaseline* detector and our *IterativeHardPositives+* detector, as seen in Figure 5.7. We also compare our technique to some state-of-the-art face detection algorithms based on neural networks like FastRCNN [51], ConvNet3D [62] and MultiTaskCNN [125]. This clearly shows that we already close the gap between cascade classifiers and neural networks a lot, while still having room for improvement.

Influence of adaptations to processing time

One must make sure that adding all this extra training data does not make the model overly complex and slow during detection time. As shown in Table 5.1 we have only a limited increase in used features as stump classifiers, while adding 50% more valuable positive training data. Furthermore, the model complexity, expressed in number of stages, drops with our models. Since processing time is a key feature for many computer vision approaches applied in embedded systems, we take the liberty of measuring processing time over the complete FDDB test set, which can be seen in Table 5.2. We average the timings to receive a timing per image, given the average resolution of the test images is 400×300 pixels. These timings are performed on an Intel(R) Xeon(R) CPU E5-2630 v2 system set-up. Our OpenCV build is optimized using the Threading Building Blocks for parallel processing. We clearly see, although we are using more features in our model, that the processing time of our *IterativeHardPositives+* model does not exceed the processing time of the *OpenCVBaseline* model. Furthermore, if we use our *BoostedBaseline* or *IterativeHardPositives* detector, we process images remarkably faster than the *OpenCVBaseline* detector.

A visual confirmation

Figure 5.8 shows some visual detection output of our algorithm. We start by selecting a low detection certainty threshold *(Figure 5.8(a))* which clearly shows that both models are able to find faces, but immediately shows the downside of the OpenCV model, which generates a lot of false positive detections.

Table 5.2: Timing results comparing both OpenCV baseline and self-trained models for the FDDB dataset.

Model	Whole Set	Per Image
OpenCVBaseline	9 min 30 sec	0.20 sec
BoostedBaseline	6 min 8 sec	0.13 sec
IterativeHardPos	7 min 7 sec	0.15 sec
IterativeHardPos+	9 min 6 sec	0.19 sec

(a) Detection results with low detection certainty threshold.

(b) Detection results with medium detection certainty threshold.

(c) Detection results with high detection certainty threshold.

(d) Cases where both detectors fail (high certainty threshold) or where OpenCV finds a detection while we do not.

Figure 5.8: Detection results and failures on FDDB dataset for (red) *OpenCVBaseline* and (green) *IterHardPos+* model.

We increase the detection certainty threshold to a mediate level *(Figure 5.8(b))* and notice that both OpenCV and our own trained model are able to find faces, but gradually OpenCV starts to miss faces that are still detected by our model. Finally, when setting a high detection certainty threshold *(Figure 5.8(c))*, we see that OpenCV misses a lot of faces that are still found by our model. But even in the case that our model detects more faces than OpenCV we still find cases where both models fail or where OpenCV actually finds a face that our

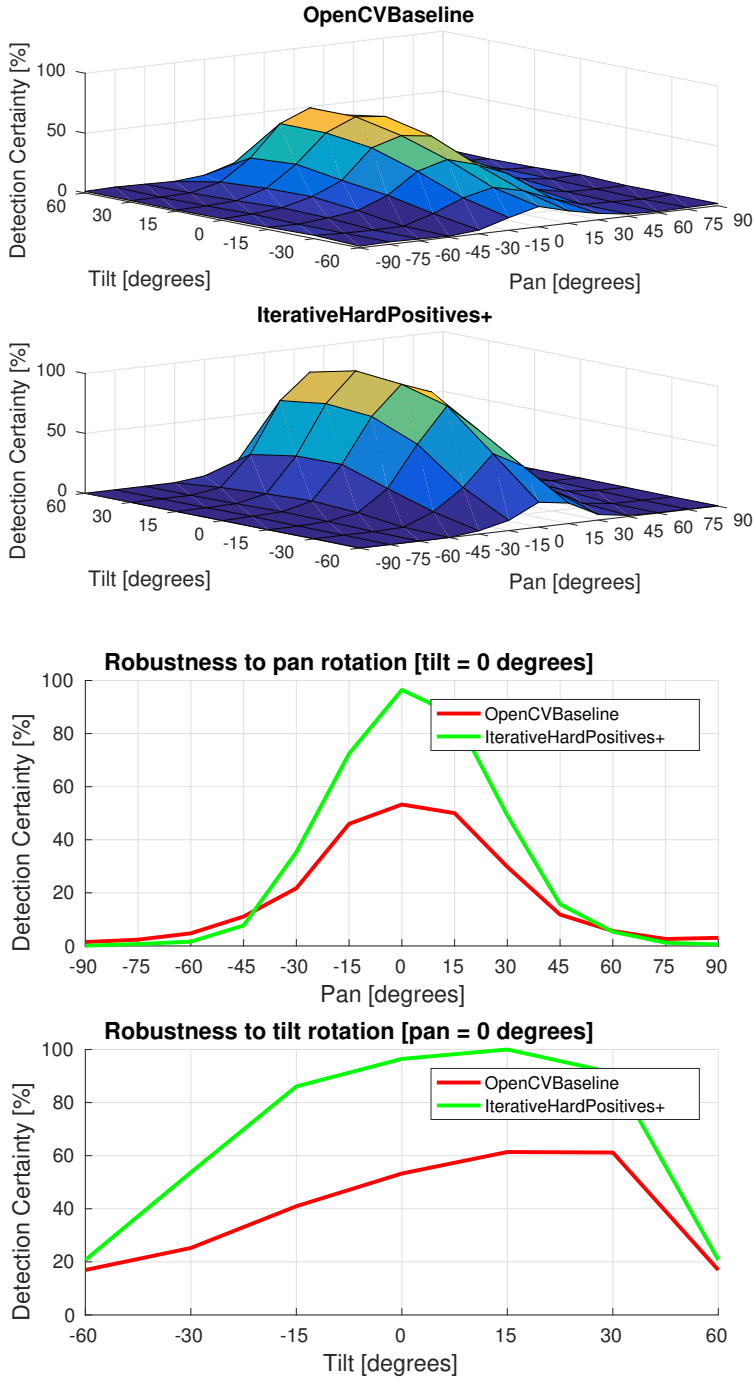


Figure 5.9: Testing out-of-plane rotational robustness for both *OpenCVBaseline* and the *IterativeHardPositives+* detector.

models do not capture, as seen in Figure 5.8(d). These undetected faces could be used as hard positive training samples but then we would need to search for a new database for evaluation purposes in order not to lose the benefits of dataset bias, where we prove that our model is independent on the dataset conditions where it was trained on.

Testing out-of-plane rotation robustness

As stated before, reducing the annotation region, which directly influences the face region that the detector will return, helps to improve the out-of-plane rotation of the face detector. To test this, we evaluated the *OpenCVBaseline* and the *IterativeHardPositives+* detector on the Head Pose Image Database [40], as seen in Figure 5.9.

This dataset contains a set of 30 sequences *(15 persons, 2 sequences per person)* where people sequentially look at different positions, each associated with a pan *(in the range [-90°,+90°])* and a tilt angle *(in the range [-60°,60°])*. At each position, we execute both detectors and return the detection certainty of the models. averaged over the 30 sequences. We use the highest returned detection score on the dataset as the outer bound of our score range and normalize all other values for this maximum. We see that in both pan and tilt angle evaluations our *IterativeHardPositives+* detector clearly outperforms the *OpenCVBaseline* detector. Especially in the tilt angle range, we see a large increase in efficiency. This extra test also confirms that at a full frontal face, the *IterativeHardPositives+* detector has about double the detection certainty as for the *OpenCVBaseline* detector, which was clearly noticeable in Figure 5.5.

5.1.5 Conclusion on active learning for object detection

In this subsection, we suggest adaptations to the current existing cascade classification pipeline in the open-source computer vision framework OpenCV with the eye on improving its frontal face detection model. We aim at reducing the huge amount of false positive detections, by guaranteeing a high precision, while maintaining the recall as high as possible, to detect as many faces as possible. We test our approach on the publicly available FDDB face dataset and prove that our adaptations to the pipeline generate an enormous increase in performance. Using our *IterativeHardPositives+* detector, we achieve an increase in recall to 68% while maintaining a high precision of 90%. Compared to a 40% precision at 40% recall for the OpenCV baseline, this is quite impressive.

The suggested adaptations to the framework and the model clearly have benefits over the currently available model. Imagine a case where the output of the face detector is used to perform face recognition. In such cases, we aim at a

precision that is as high as possible since we want to ensure that the pipeline following on the actual detection, is not provided with rubbish but with an actual face. Furthermore, our model is able to find more faces in the wild and is more robust to out-of-plane rotations compared to the OpenCV baseline model.

We should take into account that we will never hit a 100% recall on datasets like FDDB, due to some high out-of-plane rotations, as seen in Figure 5.8(d). However one could argue that faces with an out-of-plane rotation of more than 45 degrees should be found by a profile face detector and combine both detectors together, as suggested by Hulens et al. [46].

5.2 Benefits, challenges and possible expansions

The previous section discusses in detail how we implement an active learning strategy for training robust object detector with the boosted cascade of weak classifiers algorithm. We consider it important to highlight why this developed approach is useful for our specific task of industrial object detection in subsection 5.2.1. We also investigate what approaches could be used to further limit the manual input needed for building robust object detectors in 5.2.2.

5.2.1 Advantages and challenges of active learning

Active learning provides several benefits for our industrial relevant object detection applications. Besides the fact that we can already limit the number of needed training samples by building scene and application specific object detection models, and the fact that we can use scene- and application-specific constraints to reduce this even further, some of our detection models still need a couple of thousand manual annotations. Active learning proved in section 5.1 that it can literally move through several thousands of training samples and only provide those samples that actually contain an object instance that can actually introduce extra knowledge into the detection model.

Reducing the number of training samples that need manual annotation as much as possible is the biggest benefit of using active learning. In an industrial set-up, one can do the model learning overnight, or sequentially work on other tasks, and thus not lose time waiting for the actual algorithm. The active learning algorithm can collect samples on the go where it needs more direction until it reaches enough samples and then asks for limited manual input.

In this idea we can separate three different cases where the active learning algorithm needs new input:

1. An object instance is found with a small probability of being background.
2. An object instance is found with a small probability of being an object.
3. An object instance is not retrieved from the dataset but randomly selected.

The first two cases are directly related to the nature of active learning. It looks for samples that are able to define a cleaner separation plane in the high dimensional feature space. Thus low probability scoring samples are actually close to that border and could be wrongly classified. In principle we want these samples to be checked by the user. Of course, this is less invasive than asking for actual annotation because it can simply be a 'agree' or 'change' option which allows quick surfing through the generated samples.

The third case is actually a safety net we need to build in ourselves. The initial set of samples for training our first weak classifier is very critical in this process. If specific object instance conditions are not provided to the learning system, the system will be unable to incorporate this knowledge into the model. There is no way of telling how the process will react if these samples are presented in an active learning iteration. Therefore we force the system to randomly sample images from our dataset, which we annotate explicitly with a bounding box. By evaluating these extra regions, we can look for current outliers with a high probability that should, in fact, be at the other side of the separation plane.

5.2.2 Expansions

Active learning already helps to build industrial object detectors a lot. However, there are still advances in computer vision that could expand the capabilities of active learning in our perspective and that are worth investigating further.

Reinforcement learning

The biggest profit of reinforcement learning, suggested by Sutton et al. [109], compared to purely supervised learning would be that it directly incorporates the concept of exploration versus exploitation. In the explanation phase, it directly looks at current knowledge and uses that knowledge to decide on newly given samples, which is the same as in supervised learning.

However, remember the fact that we had to add manually random samples, to avoid over-fitting to the given dataset, and to explore new parts of the data that have never been seen before. Reinforcement learning implements this exploration part besides the classical exploitation approach.

Reinforcement learning is also something that is not new in the field of object detection. Peng et al. [83] already suggested using reinforcement learning for closed-loop object recognition in 1998. More recently Paletta and Pinz [81] combined reinforcement learning with active view selection for similar tasks.

A downside of reinforcement learning has been the fact that it was computationally quite heavy. However, since the rise of deep learning and affordable GPGPUs, researchers again picked up the concept of reinforcement learning for deep neural networks, like in [74, 75].

One-shot learning

The holy grail for industrial computer vision is one-shot learning, as suggested by Fei-Fei in [30]. In this work, they create a system for learning object categories based on only one to five samples per class. If we could integrate this into object detection pipelines, then this would almost completely remove the need for manual annotation. Since the rise of deep learning, this one-shot approach has gained again a lot of attention for both object recognition [53] and object detection [10] tasks. There are even several attempts at creating zero-shot learners for visual object categories [57, 105] but accuracies are far from the minimal requirements, posed by industrial object detection tasks.

5.3 Conclusion: benefits of using active learning

In this chapter, we presented an efficient solution for reducing the amount of manual work needed when training and fine-tuning object detection models to reach the high accuracy standards, by using a proposed active learning approach in section 5.1. By applying this approach we generated a new, open-source face detection model, with a higher detection accuracy than the currently embedded ones in the OpenCV framework. Since official integration into the framework, we clearly notice a higher usage of the new model compared to the older existing ones, getting a lot of positive feedback from the community on the contribution. Furthermore, we noticed several other open-source contributors took the general guidelines presented and used them to further improve other detection models in the framework.

In section 5.2.2 we discussed why active learning is actually useful for our industrial applications and we suggest techniques that need to be further investigated in the future to improve our current pipeline.

In the following chapter, we switch our baseline object detection algorithm towards a deep learning based approach. We discuss in detail the several approaches that we tried and how we tried to efficiently reduce the amount of training data needed, without sacrificing on the reported accuracy of deep learning based approaches.

Chapter 6

Usability of deep learning for industrial object detection

The work presented in this chapter will be published at the VISAPP 2018 conference [86, 92].

When we started our research, the state-of-the-art in object detection simply did not contain any deep learning approaches. The main reasons behind this were the lack of large training datasets, the availability of affordable GPGPU hardware, and the tremendous amount of time needed for training a deep learning model from scratch. These three reasons obviously made deep learning not a suitable approach for real-world industry-specific object detection tasks.

However, in latest years, the rise of affordable deep learning hardware, as well as the reduction of data actually needed to train a deep learning algorithm efficiently, changed the field of object detection completely. Nowadays it is hard to find a single object detection algorithm, in the recent academic literature that is, not using any sort of deep learning architecture.

This shift in algorithms kind of insinuated our performed research might be a waste of time. In order to be able to compare our research efforts to the current state-of-the-art in object detection, we decided to apply several deeply learned object detection algorithms on top of some new research cases. By doing so we investigate the possibility of using deep learning as an approach for industrial relevant object detection model learning, instead of sticking to older machine learning approaches like the boosted cascade of weak classifiers.

We started out by using off-the-shelf deep learned object detection architectures, but quickly noticed this only works in the case you have an object class for which the model was already trained for *(e.g. pedestrians, cars, bicycles, ...)*. In many of our industrial applications this is not the case, so using a pre-trained model is simply out of the question. Section 6.1 discusses our first experiences and contacts with deep learning in the context of industrial relevant object detection. We investigate the possibility of using transfer learning as a way of benefiting the power of pre-trained deep learning architectures while adding scene- and application-specific knowledge, generating robust and accurate deep learned object detectors, while simultaneously reducing the amount of manual input needed by the algorithm. All of this is done on top of three cases, of which one is a demo case of the game rock-paper-scissors, followed by two industrial relevant cases, one being the detection of promotion boards in stores and the other being the detection and classification of packages in supermarkets.

Section 6.2 takes a step back and investigates if deep learning is a worthy competitor to the previously used boosted cascade algorithms like the Viola&Jones and ACF approach. Furthermore, we notice that the approaches applied in section 6.1 fail on the very specific case of coconut tree detection in aerial imagery, due to the very different nature of our input data, compared to the data on which the pre-trained models are learned. We investigate the use of more pure-classification-based deep learned architectures for this problem and combine these efficiently with a multi-scale sliding window based approach to achieve our set goals.

Finally, section 6.3 draws conclusions on using deep learning for industrial object detection and makes some suggestions on promising future techniques and possible expansions.

6.1 Building robust industrial applicable object detection models using transfer learning and single-pass deep learning architectures.

Several drawbacks have kept deep learning in the background for quite a while. Until recently deep learning had a very high computational cost, due to the thousands of convolutions that had to process the input data from a pixel level to a more content based level. On high-end systems, reporting processing speeds of several seconds on VGA resolution has long been the state-of-the-art. This limits the use of these powerful deep learning architectures in industrial situations such as real-time applications and on platforms with limited resources.

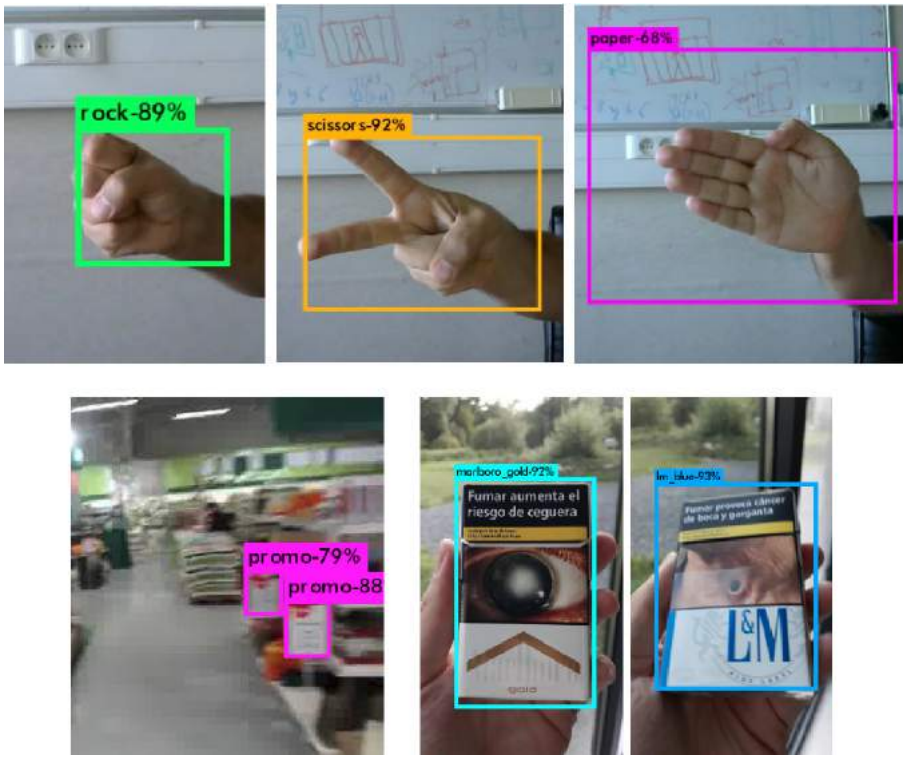


Figure 6.1: Examples of our cases: *(top)* Rock-Scissors-Paper *(bottom left)* Promotion Board *(bottom right)* Cigarette Packs.

Furthermore, deep learning needed a dedicated and expensive general purpose graphical processing unit (GPGPU) and an enormous set of manually annotated training data, both things that are almost never available in industrial applications. The frameworks for deep learning lacked documentation and guidelines, while pre-built deep learning models were not easily adaptable to new and unseen classes. Nowadays, these issues no longer exist, and deep learning has made a major shift towards usability and real-time performance. With the rise of affordable hardware, large public datasets (e.g. ImageNet [21], Microsoft COCO [66], VOT [13], Pascal VOC [29], ...), pre-built model zoos (e.g. Caffe model zoo [50]) and techniques like transfer learning, deep learning took a step closer towards actual industrial applications. Together with the explosion of available open-source, stable and well documented deep learning frameworks *(e.g. Caffe [50], Tensorflow [2], Chainer [114], MatConvNets [118], Darknet [95], ...)* this opens up a whole new world of possibilities.

Industrial object detection applications are typical cases where we have a lack of training data. Many applications do not focus on existing classes from academically provided datasets and thus do not contain publicly accessible training data. Gathering those tremendous amounts of training data *(up to millions of training samples)* and providing manual labelling, is a time-consuming and thus expensive process. Luckily transfer learning [8] provides possibilities in this case. Given a small set of manually annotated training samples, it retrains an existing off-the-shelf deep learning models and adapts its existing layer weights so that it is able to detect a completely new object class.

In this chapter, we apply this transfer learning methodology on three cases. We start with a dummy case of rock-scissors-paper *(as seen in Figure 6.1(left))* to have a look at how we can fine-tune existing deep learning models to learn a new task it has never seen before. Simultaneously we focus on using a small set of training data for the new classes, to prove we do not need huge datasets to achieve this task. At the same time, we try to ensure real-time performance, a hard constraint in many industrial applications.

This is followed by a first industrially relevant application, where we detect promotion boards in eye-tracking data, allowing companies to analyse shopping behaviour, as seen in Figure 6.1(middle). This case is used to verify the approaches of the dummy cases on top of an industrial relevant application, proving that we can achieve average precisions that can compete with our existing algorithms, without putting in too much extra effort. We investigate both the possibility of training a single class detector, as well as training multi-class detectors.

Furthermore, we explore the limits of these multi-class object detection networks, in order to be able to skip the subsequent classification step and push everything into a single combined detection and classification pipeline.

The third case, detecting and recognizing packages of warehouse products for augmented advertisement purposes *(as seen in Figure 6.1(right))*, allows us to look for the limits of using a deep learning object detection architecture for both detection and classification at once. We start by building a general cigarette box detector, followed by a 14-class detection model which directly detects and classifies the 14 different kinds of cigarette brands, instead of running a separate classification pipeline after the general cigarette box detector.

The remainder of this section is organized as follows. In subsection 6.1.1 we discuss related work and highlight the state-of-the-art in deep object detection. Subsection 6.1.2 discusses the collected dataset for training and validating the object detection models. This is followed by subsection 6.1.3 discussing the selected deep learning framework and the deep learning model architecture

which we used for fine-tuning. Subsection 6.1.4 goes over each application case in detail. In subsection 6.1.5 we discuss the obtained average precisions and execution speeds. Finally, we discuss the used algorithms in subsection 6.1.6, while in subsection 6.1.7 we draw some meaningful conclusions.

6.1.1 Related work on deeply learned object detection

Since convolutional neural networks basically process image patches, doing a multi-scale and sliding window based analysis takes a lot of time, especially if image dimensions increase. Combine this with the fact that deeper networks achieve a better detection accuracy, very deep networks like *VGG19* [104] and *InceptionV3* [110] can easily take several seconds for a VGA resolution images. To increase the execution speed of these algorithms, researchers introduced the concept of region proposal techniques. These techniques are fast and lightweight filters that pre-process images, looking for regions that might have promising content. Instead of supplying the convolutional chain with multiple millions of windows, by using a naive multi-scale sliding window approach, the region proposal algorithms reduce this to only several thousand of windows. Adding an extra region proposal network improved processing time, but still introduces an extra network which can lead to an undesired overhead.

If your industrial application concerns a class that is previously trained on any given public dataset, the above techniques provide off-the-shelf solutions. However, when your object class does not come in any of the pre-trained networks, one needs all at once a huge dataset and a lot of processing time to come up with a new model. That is why Yosinski et al. [124] investigated the transferability of deep learning features and discovered that the first convolutional layers of any deep learning architecture actually stick to very general feature representations and that the actual power resides in the linking of those features by finding the correct convolutional weights for each operator. This allowed applying the concept of transfer learning [17] onto deep learning.

Basically one needs to initialize the weights of the convolutional layers that provide a general feature description, using the weights of any pre-trained model, then using a small set of annotated application specific training samples to update the weights of all layers for the new object class. Combined with data augmentation [37], a small set of these training samples can introduce enough knowledge for fine-tuning an existing deep model onto a new class.

6.1.2 Dataset and framework

For this work, we create three datasets that we use for training and evaluation of our deep learned object detector models. In all cases we provide manual annotations, allowing for qualitative evaluation using precision-recall curves in the practical industrial cases where a validation set is provided.

For our dummy case of rock-paper-scissors, we provide respectively 61, 57 and 60 training samples for fine-tuning a model into a 3-class detector. All samples are manually annotated with ground-truth labels. As validation for this dummy case, we simply use a webcam interface and visual confirmation to check whether the produced model actually does something useful and is able to classify correctly the 3 classes with minimal errors.

The first industrially relevant dataset contains videos of an eye-tracker experiment in a Belgian shop. Data is collected by normal costumers, that are approached when entering the shop and asked to go shopping, equipped with an eye-tracker to record the experience. Users are not made aware that the experiment was actually for investigating if customers notice promotion boards placed at specific locations in the shop. Two classes of promotion boards *(as seen in Figure 6.4)* are manually annotated in each frame of the eye-tracker videos, providing robust ground truth data. We separate the obtained dataset into training and validation data, as seen in Table 6.1. For the training data, specific frames containing the actual advertisement board are manually extracted and annotated. For validation data, from 2 remaining videos frames are captured at a one-second interval, and manually annotated for validation purposes.

The second industrially relevant dataset contains 376 images of cigarette packages. In each image, the location of the cigarette package is manually annotated using a rectangular bounding box. In total 14 brands of cigarette classes are included, allowing us to both provide labels of a cigarette box and the associated brand, useful for training multi-class object detection models. On top of this training dataset, a separate validation dataset is generated, existing of 26 videos (@30FPS) of around 15 seconds, containing different views of similar cigarette packages. In total 4.279 images are added to the validation dataset.

Table 6.1: Number of promotion board samples per class.

Label	Training	Validation
small_sign	75	420
large_sign	65	960
TOTAL:	**140**	**1380**

Table 6.2: Number of cigarette samples per brand.

Label	Training	Validation
marlboro_red	63	163
marlboro_blue	14	149
marlboro_gold	29	157
marlboro_touch	14	167
chesterfield_red	75	157
chesterfield_blue	15	222
heets_gold	15	185
heets_red	15	188
heets_blue	15	229
lm_red	46	183
lm_blue	30	146
lm_gold	15	179
philip_morris_yellow	15	178
philip_morris_red	15	179
TOTAL:	**376**	**4279**

Table 6.2 contains a detailed overview of the specific amounts of training and validation samples per cigarette brand.

As deep learning framework, we decide to use Darknet [95], a lightweight deep learning library, based on C and CUDA, which achieves state-of-the-art performance with the YOLOv2 architecture [97] on well known public datasets for object detection. Moreover, it reaches real-time processing speeds raising up to 120 FPS at limited resolutions.

Furthermore, the framework allows for nice integration with our existing C++ based software platform based on OpenCV3.2 [12]. The YOLOv2 architecture does not require explicit negative training samples in order to be able to train an object detection model. It uses areas in the image that are not labelled as an object as negative training data automatically, by requiring a class probability of 0% for those pixels. However one can add explicit hard-negative training data if desired, by simply adding unannotated training images to the training dataset. How this influences the training and average precision is discussed in subsection 6.1.7.

Since the existing Darknet framework seems to be missing some functionality that we require to be able to robustly evaluate our object detection models, we build our customized version of the framework, which is publicly available at `https://gitlab.com/EAVISE/darknet`.

Our biggest adaptations are the possibility of extracting specific output formats, generating precision-recall data points and allowing to validate models with a variable score threshold. Furthermore, several small changes are made to make the output result visually more pleasing, like adding the probability scores next to the detection labels.

6.1.3 Used approach

One thing that immediately comes to mind when talking about deep learning, is that deep learning algorithms need giant amounts of training data and a lot of processing time to be able to converge to an optimal configuration. For the suggested deep learning architecture, the YOLOv2 architecture, this is also the case. The default network configuration needs 800.000 floating point operations per given image. To optimize the weights assigned to each operation, one literally needs millions of training samples. For industrial applications this is not manageable, certainly if, for each new object detection model, you need new and annotated training data.

By making use of transfer learning, we only need a small amount of application-specific training data to use the incorporated knowledge of existing deep learning architectures, trained for a specific dataset, to be able to create a new case specific object detection model. For our object detection models, we start from a YOLOv2 architecture *(as seen in Figure 6.2)*, previously trained on the Pascal VOC 2007 dataset. To be able to fine-tune this model onto a new object class several adaptations had to be made to the network.

1. First of all, we need to physically change the architecture of the network. To make sure the convolutional layers output the correct format for the detection layer, we need to adapt the total number of filters in the last convolutional layer *(the orange coloured convolutional layer in Figure 6.2)* equalling $(N_{classes} + 5) \times N_{anchors}$. In doing so we ensure that the detection layer is able to convert the final layer activations into useful detections for our case specific problem. In the case of a three-class detector, with the default number of five anchors, this would mean the number of filters of the last convolutional layer should be changed to 40 to be able to generate a correct detection output.

2. Secondly, we need to adapt the anchor ratios to our class-specific problem. These anchor ratios are extracted from our training data and represent the possible rectangle ratio dimensions a bounding box can have. This ensures that in the detection layer, prediction boxes are generated fitted in a ratio that agrees to our actual fine-tuning data from our new object

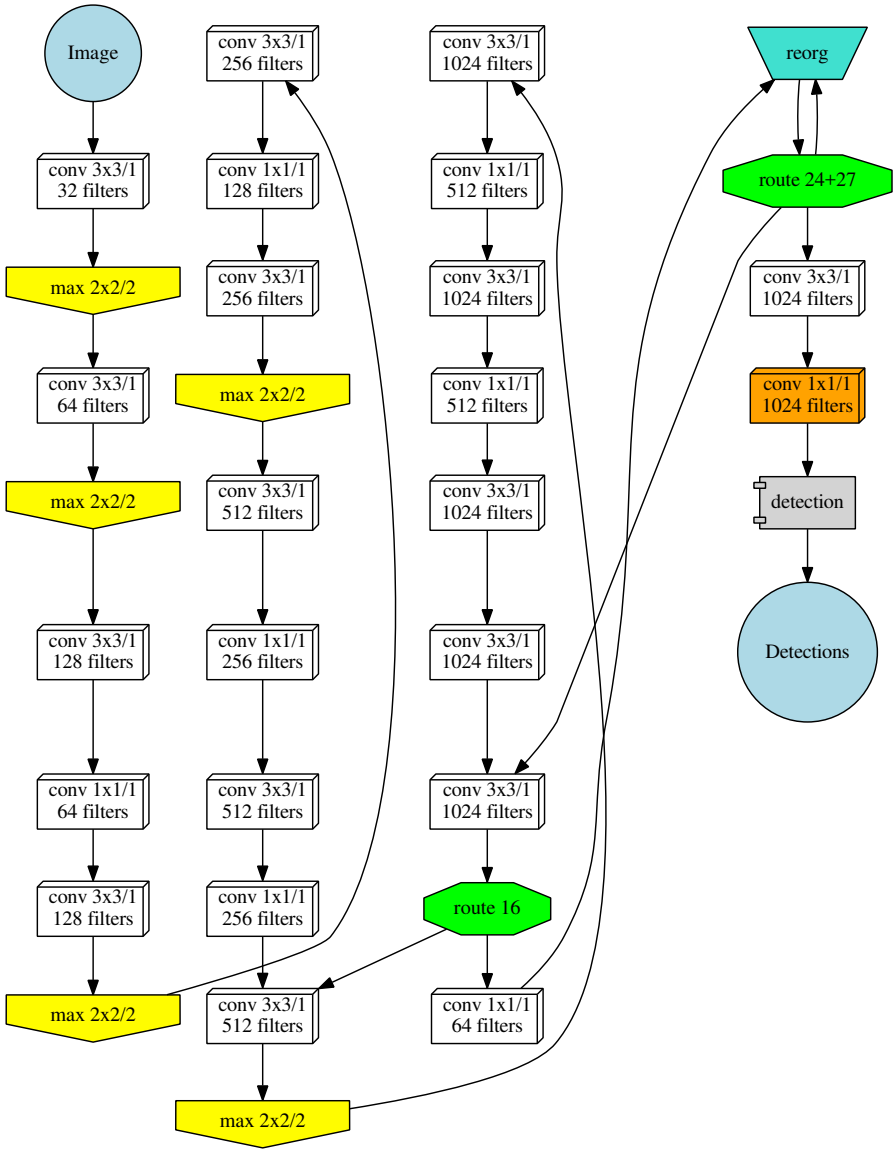


Figure 6.2: YOLOv2 architecture visualised with its different layers and in-between layer connections.

class. By default YOLOv2 uses five different ratios, to allow for slight variations due to natural variance, viewpoint deformation, ... This value should be lowered to one anchor ratio if we desire a fixed bounding box ratio and can be increased if the variance in the data increases also.

3. We need to grab the pre-trained weights from training the architecture on the PASCAL VOC 2007 dataset. We take the weights of all except the final convolutional layers and leave the weights of the final convolutional layer and the detection layer uninitialized because we changed the architecture of those and desire to train class specific weights for these.

Keeping these changes in mind, we train a new object model for our new object class, using the concept of transfer learning. We allow the weights of all layers to update *(including the ones assigned weights of the pre-trained model)* and thus converging towards the optimal solution given our new training data. This is, of course, a process that is slower than simply freezing layers, but the longer training time results in a more precise model. By using these pre-trained weights we assume that the model already learned some useful feature descriptions and constellations from the more general dataset, that will also be useful for training our application- and scene-specific object model.

Aside from using the power of a previously trained model, we also use the power of data augmentation to ensure we create a robust object detection model. A model generally fine-tunes for multiple thousands of iterations, providing a batch of 64 samples at each iteration to the model for updating the weights and reducing the loss rate on those given samples. Imagine our model would train for 10.000 iterations, this would mean the model needs more than 640.000 training samples. Given we only have around 400 training samples available this would mean that each sample is shown over a thousand times to the network, risking to completely over-fit the model to the training data. Data augmentation applies specific transformations to the image to generate a large set of data from a small set of training samples. During our training we allow the following augmentation:

- A random rescaling of the input size of the first convolutional layer, making the detector more robust for multi-scale detections. This is limited to 30% of up- and downscaling in relation to the default input layer size of 416×416 pixels.
- With each training sample, the algorithm randomly decides to flip around the vertical axis.

- The algorithm itself already transforms input images also to the HSV colour space and allows to variate the hue value. We allow a 10% deviation from the input value.
- The average saturation and exposure of the input image can deviate 50% from the input value.
- We allow the annotations of the training samples to jitter for 20% in relation to the original size. This means we can adapt the bounding box of the object annotation, as long as the cropped or moved annotation still contains 80% of the original annotation area. Besides adding training data, this also ensures a more robust model against partial occlusion and multi-scale detection.

In general, we notice that any object model we train is able to converge towards a stable model overnight, maximally taking a full day, on a Titan X (Pascal) and with a default architecture input resolution of 416×416 pixels. We halt the training when the loss-rate on the provided training samples seems to drop under 0.5, which given the initial tests with deep learning we did, seems to yield models with promising visual results, without over-fitting on the training data. That reference model can then be used to further look around that point if a better configuration might exist, as shown in section 6.1.5. Continuing the training from that point, let's say for example up to 50.000 iterations, introduces training data over-fitting and thus a model that loses its generalization properties. Keep in mind however that this is a fine-tuned model forced to work in a specific application, and thus the reported efficiencies from subsection 6.1.5 are not guaranteed in a completely different context.

6.1.4 Practical cases

Before we move on to the qualitative and quantitative evaluation of our object models in subsection 6.1.5, we elaborate on how we built object detection models for our cases. We also elaborate our choices in specific configuration set-ups.

Rock-scissors-paper

Rock-scissors-paper is a simple hand based game where two opponents count to three and then use their hand to execute one of the three hand-based motions. A fist represents a rock, a flat palm represents paper, while two V-shaped fingers represent scissors, as seen in Figure 6.3. We train a three-class model that is able to robustly separate these classes in a lab-conditioned webcam stream, meaning we have a limited amount of possible background clutter.

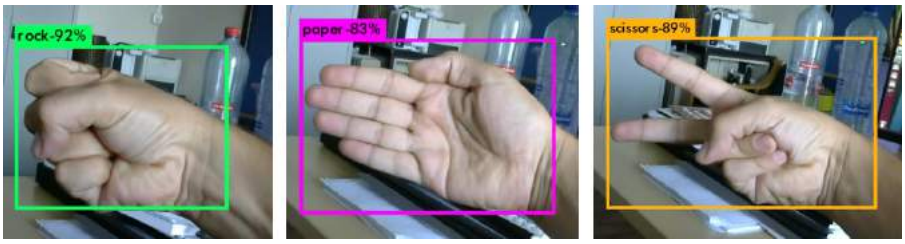


Figure 6.3: Rock-paper-scissors: an example of hand gestures.

By doing so we prove that given a limited set of training data *(only ±60 samples per class)* we are able to fine-tune an existing deep learning model on top of a new multi-class object detection task.

Detecting promotion boards in shops

For our promotion boards, we look into deep learning for two specific reasons:

1. We want to analyse complete eye-tracker experiments, removing the need for complete manual annotation of what is being looked at in the videos, linking detections to the actual eye-tracking data.
2. We want to reduce the number of annotations as much as possible without losing too much average precision on the obtained object detector.

Figure 6.4: Two promotion board classes *(left)* small_sign *(right)* large_sign.

As discussed in subsection 6.1.2 we manually annotate a small set of training samples, which are then passed to our deep learning fine-tuning system. Using this limited training set, we build two application specific object detection models. First, we build a single class detection model, able to detect the small red promotion board as seen in Figure 6.4(left). However, since the same shop also contains a second board of interest, seen as the large red promotion board in Figure 6.4(right), we explore the possibilities of building a two-class detector, able to detect both signs at the same time, giving us both a localization and a classification in a single run through the deep learning pipeline.

Detecting and classifying warehouse product packages

Our second application is the robust detection and classification of warehouse product packages for augmented advertisements. In this application, it is our task to firstly robustly locate the packages in any given input image *(as seen in Figure 6.5)*, but if possible we would like to try and classify each specific brand in the same run of our deep learning classifier. We investigate the possibilities using the cigarette package class, subdivided into fourteen separate brands.

The packages all have general cigarette box properties, while the print on the box distinguishes the brand. Therefore we start with building a robust cigarette package detector, able of localizing packages with a very high average precision in newly provided input images. Once we succeed in building a robust cigarette pack detector, we try pushing our luck and generate a fourteen-class cigarette pack detector, which is able to both localize and classify the cigarette packs in a single run, completely removing the need of a separate classification pipeline. The difference in both approaches can be seen in Figure 6.6.

Figure 6.5: Example images of warehouse products, in this case, cigarette packages of different brands under different lighting conditions.

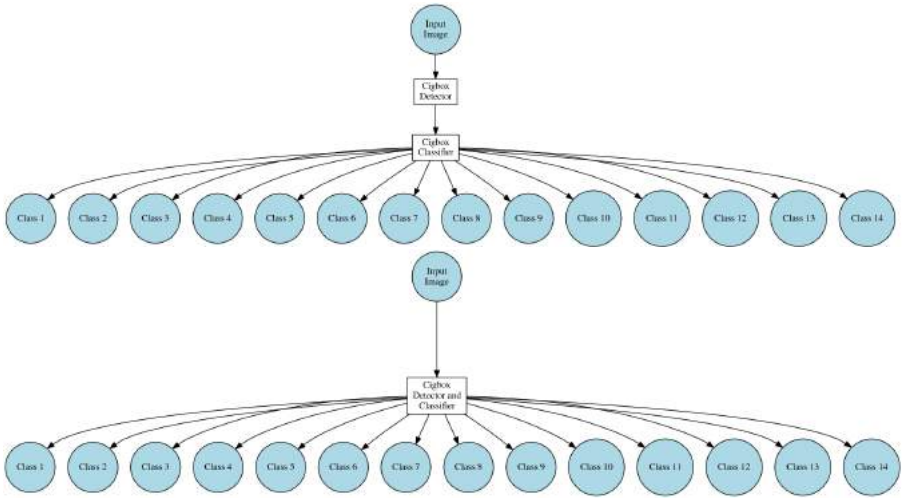


Figure 6.6: Separate versus combined processing pipeline: *(top)* Detection and classification as separate steps *(bottom)* Detection and classification steps combined.

6.1.5 Experiments and results

This section discusses in more detail the achieved results with each trained object detector. Furthermore, it highlights some of the issues we have when training these object detection models and discusses some of the limits of using the YOLOv2 architecture.

Since all our models are fine-tuned from the same YOLOv2 architecture trained on Pascal VOC 2007 [29], we have an equal size for each model. The actual model size for storing the weights of the convolutional layers is 270MB, while the GPU memory footprint of our model equals about 400MB.

Stopping the pre-trained model fine-tuning on the given dataset

When fine-tuning our new object models on top of the existing YOLOv2 detection model for the Pascal VOC 2007 dataset, we need a way of defining when our training algorithm reaches an optimal solution.

At the same time, we need to avoid over-fitting the model to the training data, keeping a model that generalizes well to never seen before data. In order to check the training progress, we continuously monitor the loss rate on the given training data. This is monitored for every new batch of samples given to the training algorithm, together with the average loss rate over all batches.

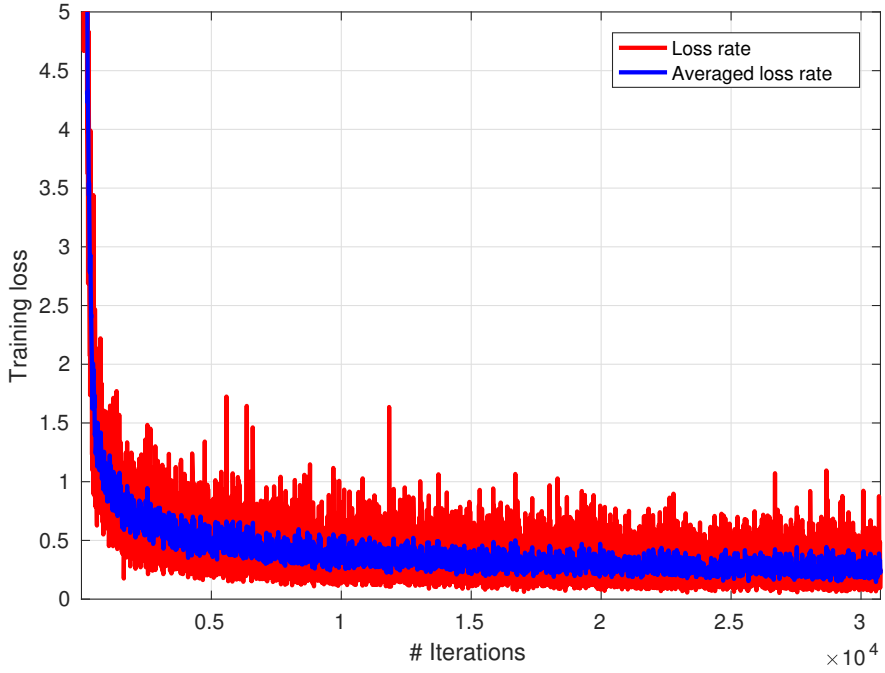
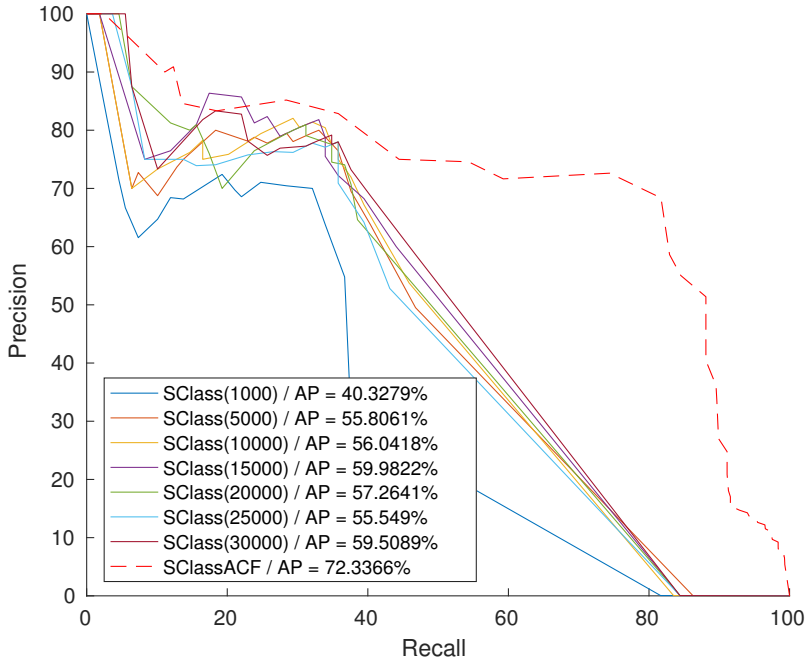


Figure 6.7: An example loss rate and average loss rate curve for single class advertisement board detector, in function of the number of iterations *(with 64 sample batches per iteration, subdivided into 8 passes).*

Once the average loss rate no longer drops, we halt the training process. Continuing the model training would result in a model that drops very little in loss rate on the training set, but which would increase in loss rate when evaluated on a given validation set.

An example of tracking the loss rate for one of our fine-tuned models can be seen in Figure 6.7. In general, we notice that a loss rate just below 0.5 seems optimal for any model, depending on the model context, yielding top-notch results when generalizing over a validation set.



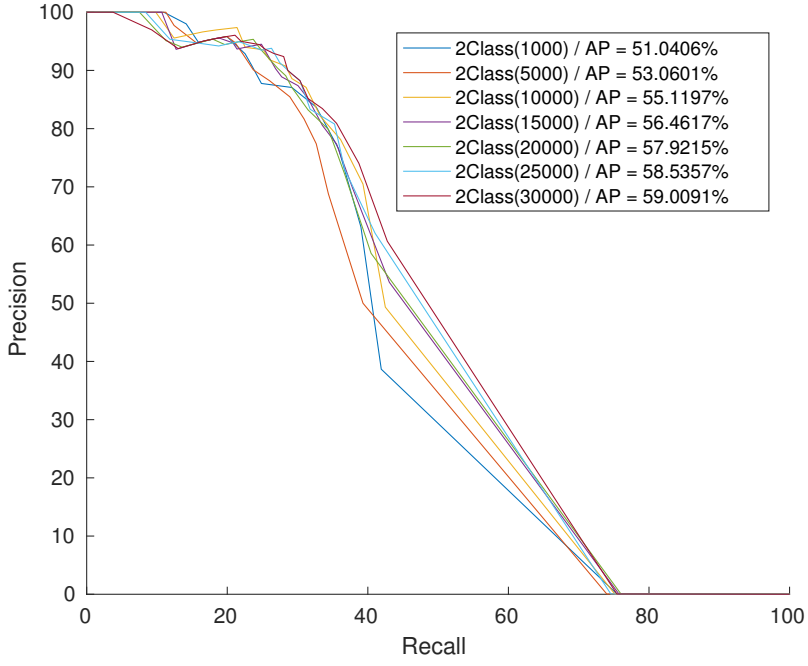


Figure 6.8: Precision-recall curves for the promotion boards single class *(top)* and the two-class *(bottom)* object detection models.

Evaluating the rock-scissors-paper detector as proof-of-concept

Our rock-scissors-paper detect achieves an optimal point at 20.000 iterations *(considering batches of 64 samples per iteration)*. Training of the model on an NVIDIA TitanX took around 6 hours, while the detector itself works at 36 FPS for a 1600×1200 pixel resolution on the same hardware. Reducing the resolution by half already pushes the detector to 60 FPS. Finally resizing the image to the dimensions of the first convolution layer, being 416×416 pixels, results in a new speed-up, achieving 90FPS. A video can be found at `https://youtu.be/VqN8XBT_q2o`.

Evaluating the promotion board detectors

Figure 6.8(top) and 6.8(bottom) show the precision-recall curves for the promotion board models. For both models, we run the training overnight halting the training at 30.000 iterations. We evaluate the precision, recall and average precision performance of each detection model at 1.000, 5.000, 10.000, 15.000, 20.000, 25.000 and 30.000 training iterations.

Initially, we see an increase in average precision when raising the number of iterations for each model. However, once the average precision *(calculated on the validation set)* starts dropping and the loss rate of the model on the validation set increases again, we select the previous model as best fit, in order to keep the model that generalizes best on the given validation set.

For the single-class detector, this means a model of 15.000 iterations at an average precision of 59.98% while for the two-class detector we select the model of 30.000 iterations at an average precision of 59%. Looking at these curves it seems that the two-class model might still be converging to an optimal solution and thus continuing training with more iterations might be worthwhile.

We notice that our models seem to be under-performing, seeing their optimal configuration only achieves a 65% precision for a 50% recall. Several reasons for this can be found. First of all we are aware that the eye-tracker data suffers a lot from motion blur in the validation set while there is no motion blur in the training set. Secondly, we know that the YOLOv2 architecture is only able to detect the range of object scales it has seen at training time. Several of the validation scales, annotated by a human annotator are different from the training size, mostly due to promotion boards further away or closer by to the customer wearing the eye-tracker than during the training data capturing. Finally, YOLOv2 has problems with detecting objects that are relatively small in comparison with the image dimensions. While this is not an issue for a human annotator, who carefully selects these boards using the human vision system, the author of the YOLOv2 algorithm already discusses that this remains an issue for grid-based single shot deep learning architectures.

However having these precision and recall values is actually more than enough, given the context where we are using these detectors. We only need to signal when a customer has seen a promotion board, related to the gaze-cursor location. If we are not able to find a promotion board on a smaller scale, once the customer comes closer to the sign, we are able to robustly detect it with a high probability of detection. It is thus always important to put these precision-recall values in perspective within the context of the application.

Looking at the execution speeds of the trained models, we notice that both the single class and the two-class model perform around 55 FPS on an NVIDIA TitanX for a 1.280×720 pixel resolution. A video of the single class promotion board detector can be found at `https://youtu.be/dQIdRSDm6Jc`. Again we guarantee a real-time performance on full resolution images, which can increase if resolution decreases.

The same case was also processed using an ACF model trained on the same training data. To compare this algorithm to the obtained deep learned models, we plotted the precision-recall curve of the ACF model in Figure 6.8(top). We clearly see that ACF outperforms our deep learned models with an average precision of 72.33% for this case, but we already discussed previously why our deep learned models seem to fail in this case.

Evaluating the cigarette package detectors

For our general single-class cigarette pack detector, we perform model fine-tuning for 5.000 iterations, while for our fourteen-class cigarette pack is trained for 15.000 iterations. Both of these settings lead to a convergence in loss rate, just below the 0.5 threshold, as previously defined as an optimal setting with our first industrial relevant application.

Figure 6.9 gives us an overview of the produced precision-recall curves for the trained models. At the bottom left part of the figure we observe the performance on the complete validation dataset *(so all fourteen brands together).*

Our general single-class cigarette box detector is able to robustly detect every single object instance in the large validation set, obtaining an average precision of 100% and thus being the optimal solution for the task of accurately localizing cigarette boxes. The fourteen-class object detection model is as promising, obtaining an overall average precision of 99.87%, however directly providing the correct brand label in combination with an accurate localisation.

Further investigation in the small accuracy difference between both models indicates that the Heets Gold brand is the sole cause, not able to reach a class-specific average precision of 100% like all other brands, only obtaining a 95.66% average precision.

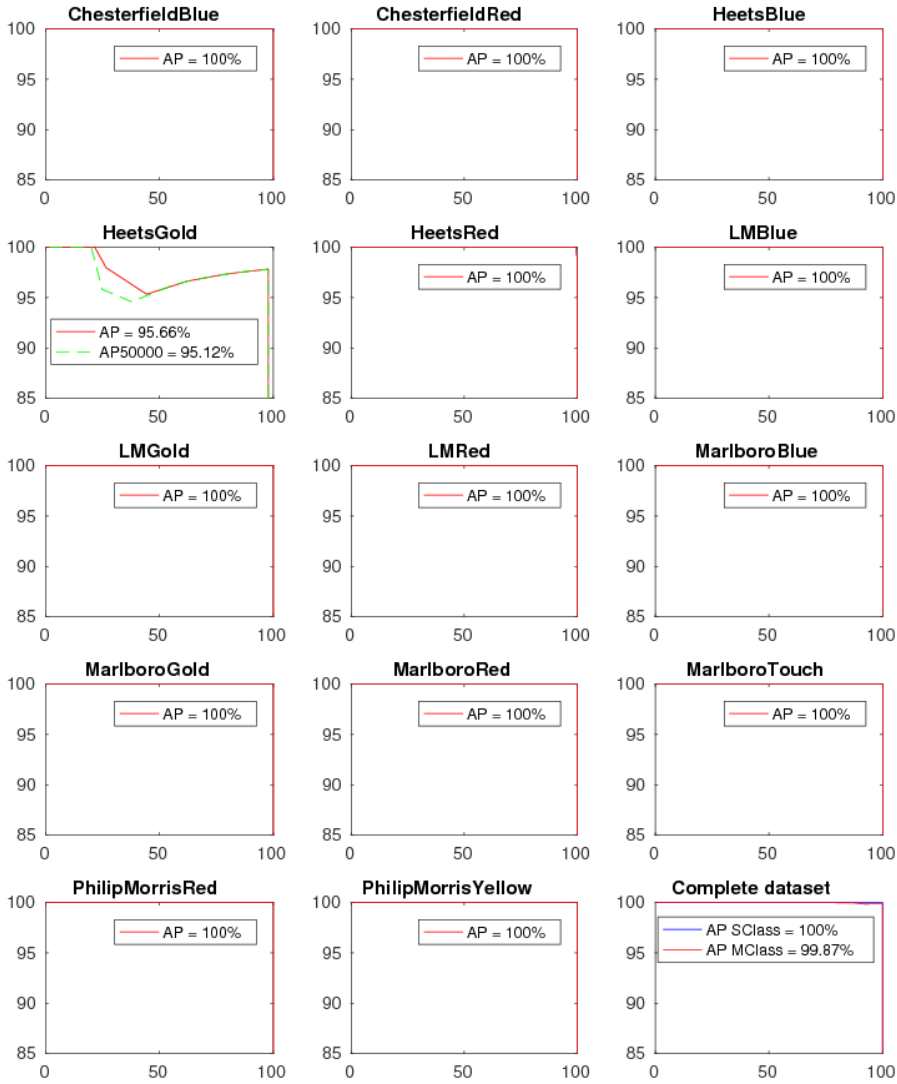


Figure 6.9: Precision-recall curves for our cigarette brand case, showing a per brand evaluation, a combined evaluation and a small extra investigation for the HeetsGold brand.

Training the model further to 50000 iterations, in the hope this class also converges to 100% average precision failed, rather leading to over-fitting of the training data, noticeable in the drop in average precision.

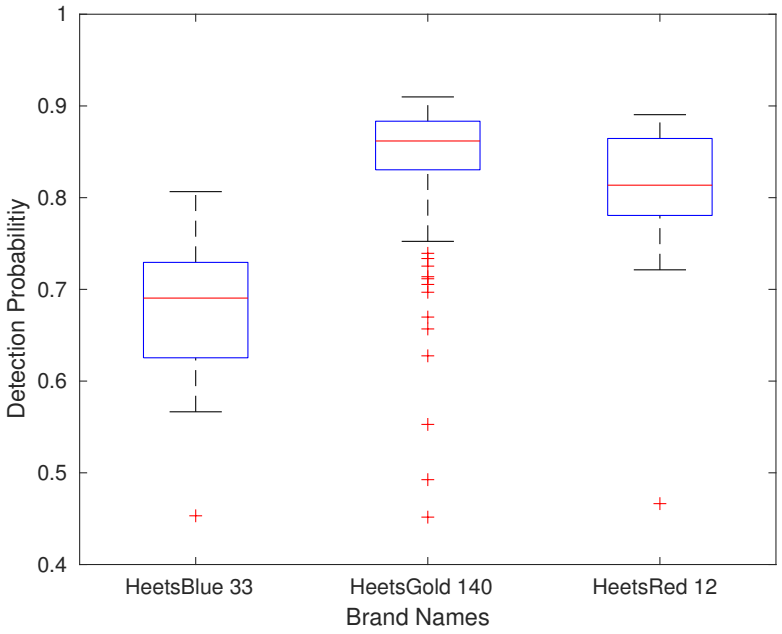


Figure 6.10: Box-plot of the Heets brands probability scores.

As a deeper investigation into the issue with the Heets Gold class, we compared the performance of the three different Heats classes *(Blue, Red and Gold)*, as seen in Figure 6.10, evaluated on validation data only containing Heets Gold cigarette packages with the goal of uncovering where it goes wrong. The detections are processed at the 0.4 probability threshold which seems the optimal threshold for the Heets Gold class, given its precision-recall curve.

This clearly shows that given only Heets Gold data for validation, there is still a large amount of data that gets classified as Heets Red or Heets Blue, and this with a high class-probability. All other brand classes were discarded from the box-plot because they did not trigger meaningful detections given the 0.4 probability threshold.

When visually inspecting the Heets class this behaviour is kind of expected, and actually, the detectors are already performing better than expected, since there are only very subtle visual changes between these three sub-brands, as seen in Figure 6.11. Even a person labelling a validation set can easily make mistakes in these classes, which have limited visual differences.

Figure 6.11: Examples of the different Heets brands: *(left)* Heets Blue, *(middle)* Heets Red, *(right)* Heets Gold.

Keeping in mind that our network resizes the input data to a fixed 416×416 resolution *(on which the model was trained to fit into GPU memory during training)* this can lead to information loss of these very small details. This results in a deep learning architecture that is even more challenged in finding an optimal separation between the sub-brand classes. Given that this architecture is basically a localiser and not a classifier, and that a human annotator makes equally similar mistakes, we decided that the lower average precision of 95.66% for this single class is actually acceptable.

Looking at the execution speeds of the trained models, we notice that both the single-class and the fourteen-class cigarette package detector are running at 70 FPS for a 720×1.280 pixel resolution on an NVIDIA TitanX. Some sample validation frames of the generic single class cigbox detector can be seen in Figure 6.12, while a selection of validation frames of our brand-specific cigarette package detector can be seen in Figure 6.13.

Figure 6.12: Example validation frames for the single-class cigbox detector. A video of the complete validation set can be found at `https://youtu.be/mwR6duCwKXw`.

Figure 6.13: Example validation frames for the multi-class brand specific detector. A video of the complete validation set can be found at `https://youtu.be/qqdbythvgTE`.

6.1.6 Discussion on using transfer-learned deep learning for industrial applications

In this subsection, we want to discuss some restrictions of the current YOLOv2 framework, shed a light on some issues that still arise when using our models and finally draw some conclusions on our used approach. Finally, we suggest ways for further improving the current pipeline.

Drawbacks using YOLOv2

We have to keep in mind that using the YOLOv2 architecture does have some drawbacks. Due to chosen internal construction of convolutional layers and the final detection layer where YOLO is combining its probability maps, it is unable to robustly detect objects that are small in relation to the size of the input frame. This is something that happens quite frequently in our promotion board detection case, resulting in seemingly worse detection results. However, the research of Tijtgat et al. [112] already proves that resizing the input layer of the deep learned network helps in detecting smaller objects in the image, increasing the average precision of the detector, however at the same time increasing processing time.

Other deep learning architectures or other trained object detection models might have fewer problems with this. Looking at our cigarette packages detection pipeline, once the object instances cover more than 20-30% of the total image pixels, this issue is completely gone.

Due to the way YOLOv2 groups its activations, neighbouring detections, get frequently melted together in a single detection, which makes precision-recall evaluations looking worse since several objects get localised as one detection.

Figure 6.14: Detection triggered by our multi-class cigbox model on general rectangular boxes, like a box of chocolate chips.

The architecture has a non-maxima suppression parameter that can be tweaked for this, but we think investigating this further is out of the scope here.

Finally, any deep learning separates given training data based on a combination of convolution filters, trying to find an optimal solution. However, we do not tell it how it should do its task, which sometimes has undesired effects. For example in the case of our general and multi-class cigarette box detector, when presented a similar rectangular shape *(e.g. a box of chocolate chips)* the model also triggers a detection, as seen in Figure 6.14. This is normal since a cigarette package is first off all a rectangular box. Still, it is undesired that other boxes trigger similar detections.

Avoiding detections on general rectangular shapes

To remove the effect of other rectangular packages triggering the cigarette package detector, we investigate the detection probability range. Given a set of cigarette packs and some random other packs *(containing both random market packages and other cigarette brands)*, we noticed that there is a large probability gap between known and unknown packages. Simply placing a probability threshold of 65% already results in the removal of all detections on non-cigarette packs and other brands on cigarette packages.

We are interested in other possible solutions to this problem, that do not include setting a specific hard threshold on the probability output since new unseen data could screw up this approach and yield probabilities for actual objects that are lower than the given score.

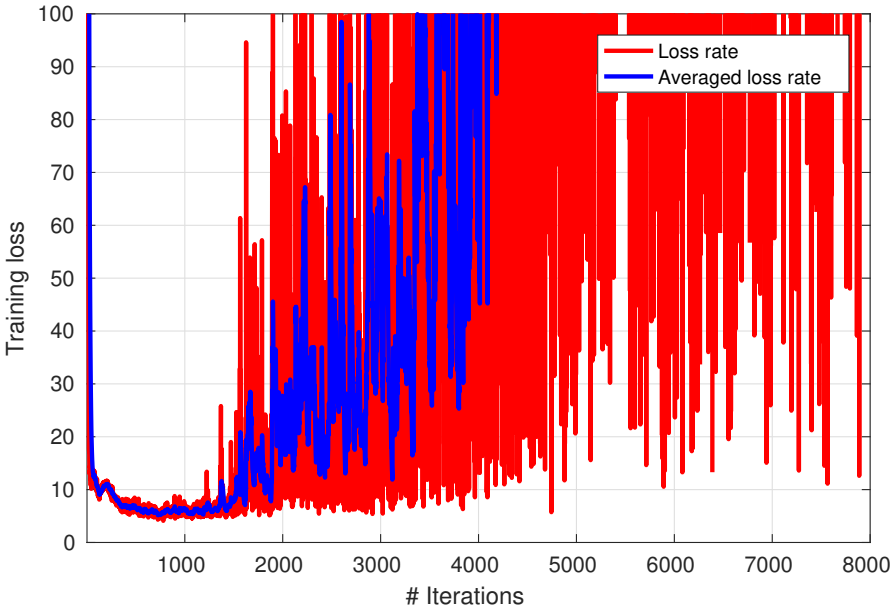


Figure 6.15: Training a cigarette box detector with a generic box as an extra class, leading to a loss rate explosion.

1. Adding a negative class to the multi-class detector

Our first attempt exists of adding an extra class containing all negative rectangular object candidates we do not want to detect. By doing this we basically want to force the deep learning network to learn all cigarette classes and on top of that an extra general box detector, which gives a more generic label that can be ignored. For this, we add extra negative training data, collected from twelve household rectangular boxes and started training again.

Figure 6.15 shows that the initial iterations show a promising drop in loss rate, however, at a certain moment we experience a training loss rate explosion, and the model is not able to recover from this.

An explanation can be found in the network architecture. First of all, we force our architecture to learn possible backgrounds *(seen as negatives)* as an actual class. The variation in those generic boxes is so large, so it will be very difficult to converge to a low loss rate. Secondly, the network architecture does not allow to learn more than 1.024 different objects as a single class, which is based on the final convolutional layer. We could increase these dimensions, but this will make a more complex model and will have no guarantee of success.

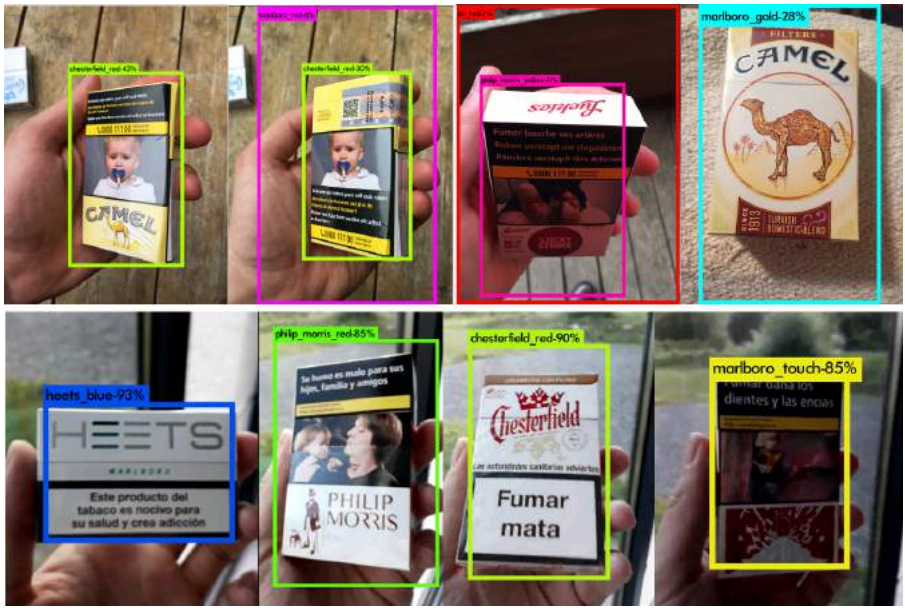


Figure 6.16: *(top)* Examples of classes that are unseen and untrained, yielding clearly lower probabilities than *(bottom)* classes that have been trained, allowing for easier separation between allowed and unallowed detections based on class probability.

Furthermore, the training of the model would possibly no longer fit in our GPU memory. Simply said if we label many different objects as one class named 'other_packages', then the negative class will occupy more weights of the resulting network, and thus the remaining objects will have less weight for memorizing their object properties. This, in turn, will greatly worsen the memorization of the classes we need.

2. Adding hard negative training images

A second attempt is made by adding hard negative training images, images containing generic boxes that are not annotated, without changing the number of classes to learn. The architecture will force the learning process to try and make the response of those hard negative images zero and thus hopefully improve the ignoring power of the model for generic packages.

This approach works and yields a fine-tuned model with a low loss-rate. The influence of this hard negative data is not removing the generic box detection properties from the model, which it still needs for detecting the actual classes, but when looking at the probabilities for the classes, we clearly notice a larger gap in probability between object class instances and generic box detections, allowing us to easier set a threshold on this detection probability.

Caution should be used here. This only worked when using a limited set of hard negatives samples in relation to the number of training samples per class. Adding too much hard negative samples adds the risk of generating batches of training samples that contain almost no actual object annotations. This, in turn, forces a model to fine-tune on basically detecting nothing and ends up with a model which reaches an optimal loss rate, on exactly that task, but is unable to detect any kind of cigarette pack.

Examples of generic packages or cigarette packages of untrained brands, yielding low probabilities compared to actually trained cigarette brands can be seen in Figure 6.16. Knowing that trained classes trigger detections almost always above a probability of 85% ensures us that there is still a margin for defining the most optimal threshold.

6.1.7 Conclusion on using transfer-learned deep learning for industrial applications

We prove that using deep learning for industrial object detection applications is no longer infeasible due to computational demands and unavailable hardware. By using an off-the-shelf deep learning framework and model fine-tuning we succeed in building several robust object detection models running on the GPU. Furthermore, we have shown that this deep object detection can be applied to toy examples, e.g. the game of rock-scissors-paper, allowing to fairly easily obtain highly accurate object detectors in lab conditions without any large effort and with minimal training data.

Our application of promotion board detection proves that the complete pipeline is also valid for industrially oriented cases, using a very limited training dataset and thus minimal manual effort, yielding models that achieve moderate average precisions. At the same time we express the fact that not all cases actually need a 100% average precision model, in order to do its job, like in the case of eye-tracker analysis.

Finally, the application of cigarette package detection and classification tries to push the limits of deep learning and model fine-tuning on industrial cases.

By using again very limited datasets *(keep in mind some classes have only 15 labelled image samples)* we achieve a perfect solution for our application, achieving a 100% average precision when it comes to cigarette box detection. Furthermore, if we directly incorporate the classification in the detection pipeline, we achieve a remarkable 99.87% average precision. This perfectly proves that deep object detection could be a valid solution to several industrial challenges that are still around nowadays.

6.2 Comparing Boosted Cascades to Deep Learning Architectures for Fast and Robust Coconut Tree Detection in Aerial Images

Getting a robust and accurate location of any object in a given input image is a key part of solving many automation tasks. In most cases the localization is only a small part of the complete pipeline, thus requiring a very high accuracy, in order to reduce the propagated error as much as possible.

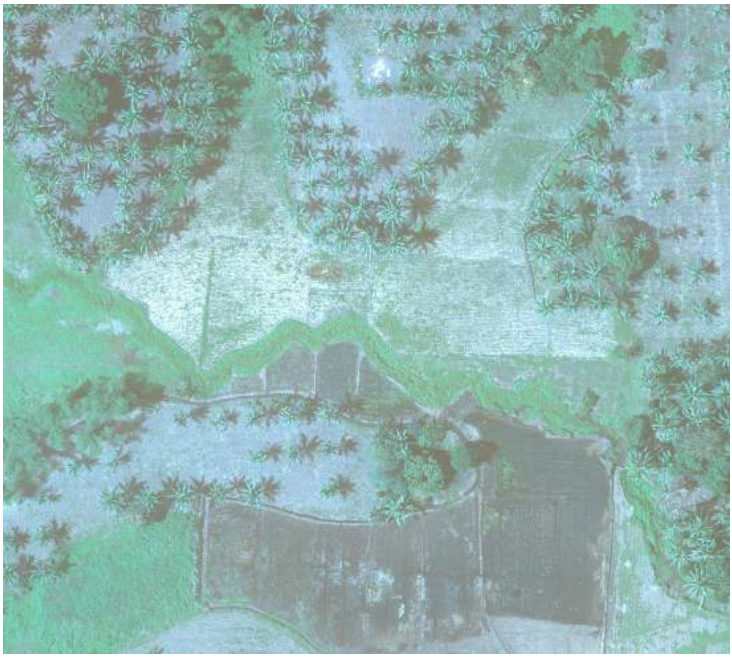


Figure 6.17: Example input image with coconut trees.

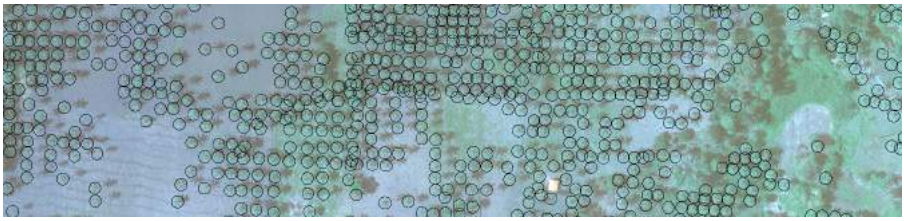


Figure 6.18: Example of an aerial image containing manually annotated coconut trees.

In the field of aerial image analysis, we notice that many tasks are still done by manual labour. Several large companies use workers in low-cost countries to manually analyse the image data and locate specific object classes in the given data. In our application of coconut detection in aerial images *(see Figure 6.17)*, the workers are asked to click on the coconut tree centres, after which a circle with a predefined diameter is placed as an annotation on top of the coconut tree, which is seen in Figure 6.18. Not only is this a costly process, it is also very cumbersome and time-consuming. Furthermore, to avoid bias introduced by a single annotator, we need to build-in redundancy, by averaging multiple annotations of the same object made by different annotators. Additionally, manual annotation is prone to mistakes when performing these repetitive tasks.

Many of these tasks could be automated given the possibilities of state-of-the-art object detection. These systems can, given pre-trained models, locate objects in the input image with human-like accuracy using the power of machine learning. Compared to human annotators these repetitive tasks are just perfect for automated systems, which can heavily parallelize these tasks and look for multiple object instances at once.

The challenging part lies in finding the correct algorithm for training these accurate machine-learned detectors. This is where we locate our research on finding the best algorithms for automated coconut tree detection. We are convinced that several proven object detection algorithms based on boosted cascades of weak classifiers [120, 26, 25] are a perfect solution for the task.

On the other hand, deep learning algorithms have introduced a new wave of state-of-the-art object detectors, capable of achieving top-notch accuracy results. Combined with the fact that the required hardware is becoming affordable and the fact that many pre-trained networks already exist, it seems a valid alternative to the cascade classifier based approaches.

In this section, we compare these well-known cascade classifier object detection algorithms to these new and powerful deep learned object detection algorithms. We evaluate the trained detection models both in achieved accuracy and execution speed, keeping in mind that achieving real-time performance is in many cases a hard constraint for actual applications. Furthermore, we give recommendations on how to efficiently use deep learning algorithms in the context of aerial image object detection and propose some general rules to keep in mind when building these models.

The remainder of this section is organized as follows. Related work on the state-of-the-art in object detection was already discussed in chapter 2. We start with section 6.2.1 where the collected data for training and validating our object detection solutions are discussed. Subsection 6.2.2 and 6.2.3 discuss the different approaches we suggest for coconut tree detection. The achieved results using these techniques are discussed in subsection 6.2.4, followed by conclusions in subsection 6.2.5.

6.2.1 Dataset and framework

As training and validation data for our used approaches, we make use of a 10.000×10.000 pixel aerial image covering a partial coconut plantation provided in RGBA format. Inside the image, all coconut trees are manually annotated using their centre position. The average size of a patch covering the whole tree is 100×100 pixels, so this size is used as annotation patch around the centre position. The image contains 3.798 coconut tree patches, while the remaining image parts are used as background information.

In all cases we split the provided data into parts, using one part for model learning and the other part for model validation, to ensure the detector is not validated on training data. Specific data splits are discussed at each technique specific subsection.

To train our object detection models we use three publicly available frameworks. The first boosted cascade approach, based on the principle of Viola&Jones with LBP features [120], is trained using the implementation of the OpenCV3.2 framework [12]. The second boosted cascade approach, based on the principle of integral channels introduced by Dollár [25], is trained using the author's MATLAB toolbox [23]. For our deeply learned models, we start by using an implementation of the *InceptionV3* architecture in Tensorflow [1], but obtained unsatisfying results using that framework. Therefore, we switched to the C and CUDA based framework Darknet [95], which includes both the classification *(Darknet19,Densenet201)* and detection architectures *(YOLOv2)* we further test in our research.

Table 6.3: Training data for the Viola&Jones based detection models and the model complexity.

	# pos	# neg	# stages
Model 1	1000	2500	16
Model 2	1000	5000	15
Model 3	1000	10000	15
Model 4	2000	8000	16

6.2.2 Approaches with boosted cascades

In this subsection, we will discuss both the different boosted cascade approaches we used for training our coconut tree detectors, and the specific amounts of training and validation data used.

Adaptive boosted cascade of weak classifiers

Our first approach is a boosted cascade of weak classifiers [120] using the adaptive boosting algorithm [68] for learning the weak classifiers, based on the local binary pattern (LBP) feature representation [4]. This invariant feature representation ignores colour information and works directly on a grayscale image, focusing on local differences in pixel intensities.

For training, we split the image into four equal parts. The annotations of the top left image part are used as positive training samples, while the remaining image parts are used for validation. As background training patches we randomly sample patches at the model size, from the image, not containing actual coconut trees. We increase the number of negative samples with each model, to obtain a more accurate detector with less false positive detections.

On top of the gathered training samples, we apply data augmentation for our final model, by randomly flipping the training patches around their vertical or horizontal axis. The number of training samples used for each model can be seen in Table 6.3, together with the number of stages of weak classifiers, indicating the model complexity. All weak classifiers are represented as single depth binary decision trees on top of the selected LBP features.

Aggregate channel features

In comparison to the Viola&Jones algorithm, the algorithm of Dollár et al. [25] proposes to add more than a single invariant feature representation to the boosting process. By adding colour, gradient filters, Gabor wavelets, ... the accuracy of the trained detectors increases compared to a single feature channel.

We first train a model using a similar amount *(2000 samples)* of positive training data to the best performing model of the previous technique. However, we note that the negative data might be gathered from patches that are also validated afterwards since the single top left corner did not contain enough background patches to use in the Viola&Jones approach. Therefore, two extra ACF models were trained, splitting the dataset into a lower *(1.741 training samples)* and an upper *(1.914 training samples)* image half. We train a model using one half and validated the model using the other half of the image. In general, the ACF algorithm uses a lot more negative training samples gathered from the same image as the positive training, leading up to 150.000 patches.

6.2.3 Approaches with deep learning

After training our boosted cascade models, we switch to the deeply learned models. We first try learning a complete model without initialized weights, then apply several transfer learning approaches, where existing weights of a pre-trained deep model are fine-tuned towards application-specific weights, resulting in a model that can detect the new object class. Finally, we investigate the difference between classification and detection architectures in deep learning and their applicability on coconut tree detection in aerial imagery.

Learning a complete new deep model

Although literature [94] advises not to train deep models if you do not have large datasets available, we train a completely new model on the available case-specific data, where no weights are initialized by using a pre-trained model. This model seems to converge, looking at the loss-rate over the number of training iterations, but in subsection 6.2.4 we discuss why this converged model is misleading.

Freezing (n-1) layers and fine-tuning final layer weights

A second approach is to freeze the weights of the pre-trained convolutional layers, and only re-train the final layer and its connections. This forces the deeply learned model to make new constellations of existing features for a new object class. We apply this approach to the existing *InceptionV3* model inside TensorFlow and try to fine-tune the final layer to be able to classify coconut trees in aerial image patches while freezing all other convolutional layers.

One major advantage of this approach is that the amount of data needed for this kind of transfer-learning is very small. Sample cases in the TensorFlow framework prove that only 75 samples per class can already be enough for satisfying results.

This approach only works if the object class to be detected is somehow related to the data contained in the initial dataset on which the model was trained. If the data is however drastically different, like in the case of aerial imagery, then obtaining satisfying results using this approach is quite hard, as illustrated in subsection 6.2.4 and other approaches should be considered.

Fine-tuning weights of all layers

Instead of freezing the weights of all the pre-trained layers, we can also tolerate slight adaptations of the pre-trained weights of the convolutional layers. This allows to change the learned features to be more specific to our desired detection task and then learn a constellation of those new fine-tuned features on top of that. When doing so, setting a small learning rate is mandatory, else the initial weights will be changed too drastically too fast, prohibiting the model to converge on an optimal solution.

Using Darknet, we apply transfer learning using this complete fine-tuning approach on both the *Darknet19* and the *Densenet201* architecture, trained on ImageNet, with the goal of obtaining a deep-learned classifier for our new case of coconut tree classification, using a smaller set of case-specific annotations.

From classification towards detection architectures

Since we are aware that using a classification network implies that we need to provide a multi-scale sliding window based approach for gathering image patches, we try training a single pass detection based model *(YOLOv2 architecture)*.

Unfortunately, due to the coarse grid-based region proposals, the proposed architecture is not able to cope with dense object-packed scenes, where object instances are closely together and slightly overlapping. This triggers detections that cover multiple object instances, instead of retrieving single object instances and furthermore doesn't allow the model to converge to an optimal configuration. This is a problem in our application of coconut tree detection in aerial imagery and thus this approach was abandoned.

6.2.4 Results

This section discusses the various results we obtain with the different object detection approaches focusing on our case of robust and accurate coconut tree detection.

Viola&Jones-based object detection

Figure 6.19 displays the obtained precision-recall curves for the Viola&Jones boosted cascades of weak classifiers using local binary pattern features. For each detector, we also report the average precision (AP), which is calculated as the area-under-the-curve for the given precision-recall curve.

The closer the precision-recall curve lies to the top right corner, the better the detector. Increasing the number of negative samples, which gives the model a better descriptive power for its negative class, seems to work well. This should in principle also mean a higher average precision, but we reckon our graph does not directly represent this. Our OpenCV based implementation does not allow to generate more precision-recall points for the given data, and since we do not want to guess the curvature, we do not take the non-existing area into account. This might give a wrong AP impression.

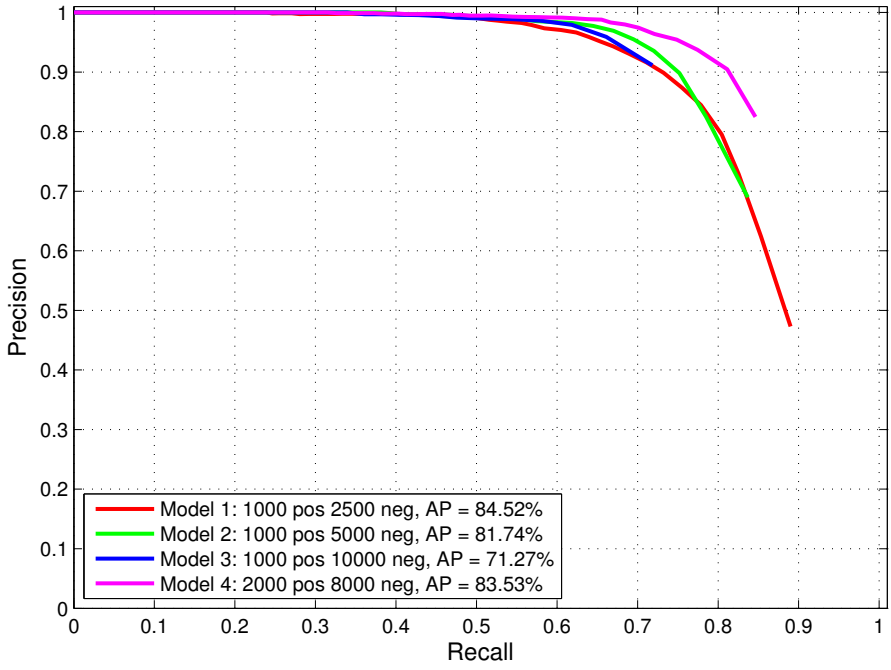


Figure 6.19: Precision-Recall curves for different Viola&Jones based detection models with the number of training samples and the average precision.

We conclude that given a fairly limited set of case-specific annotated training data, and a limited training time of only 2 hours, we obtain a detector that is able to detect coconut trees with 90% precision at a recall of 80%. Furthermore, we notice that applying data augmentation helps to boost the generalization properties of our boosted cascade models. The final model performs detections on a 10.000×10.000 pixel image within ten minutes.

ACF-based object detection

Figure 6.20 shows the obtained precision-recall curves for the ACF boosted cascades. We immediately notice that this framework is able to draw power from multiple feature channels and is thus able to obtain higher average precisions. Our best scoring model, trained on the bottom half of our dataset image and validated on the top part, achieves an average precision of 94.55%.

The optimal point of the best model, and thus the optimal setting of our detector, achieves 96% precision at a recall of 90% which is quite amazing given the very limited training time of only 30 minutes. The best performing model performs detections on the 10.000×10.000 pixel image within five minutes.

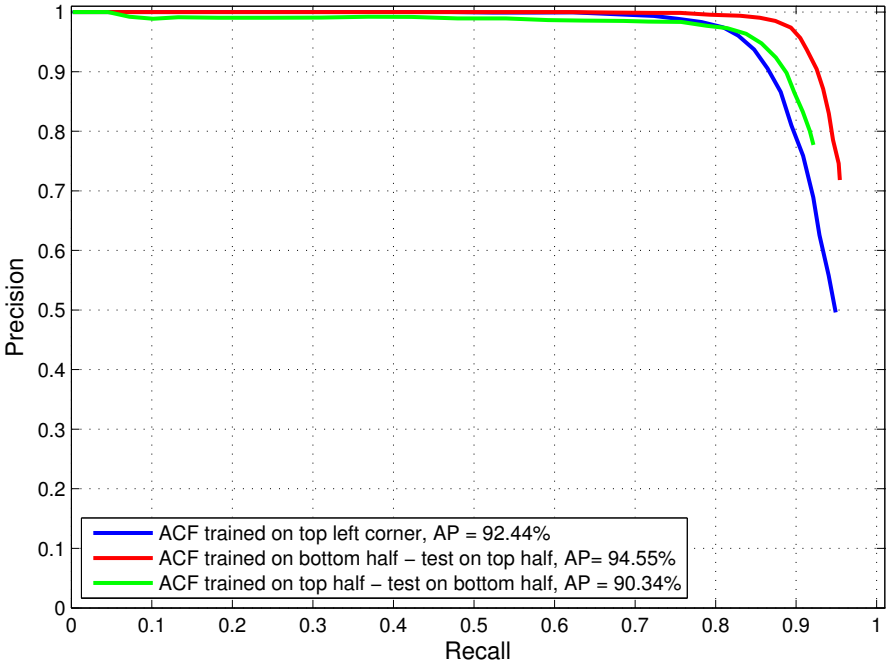


Figure 6.20: Precision-Recall curves for different ACF based detection models with their average precision.

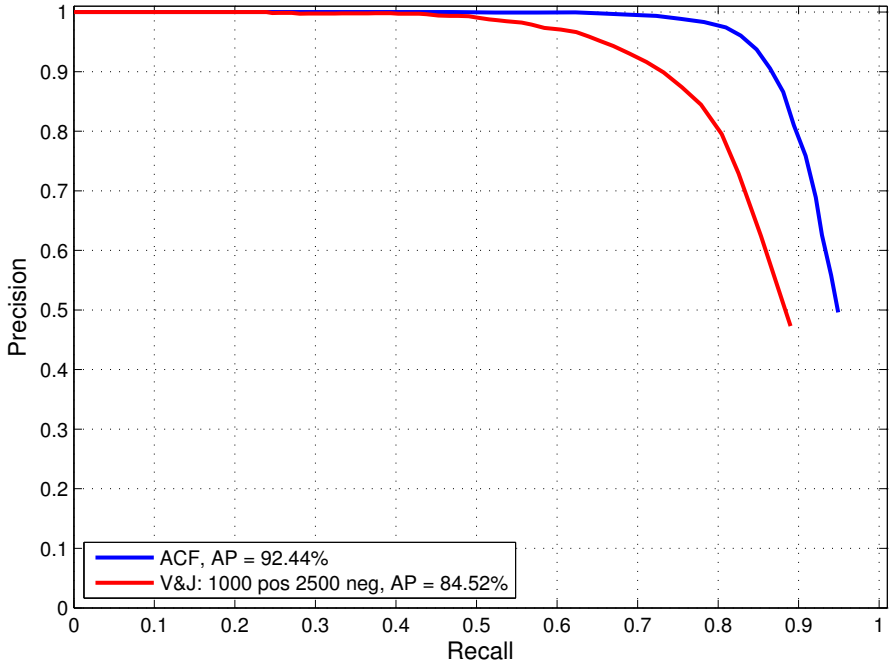


Figure 6.21: Precision-Recall comparison between the best Viola&Jones and ACF models on the same validation dataset.

Viola&Jones versus ACF

Since both detectors are validated on different amounts of data, we decide to perform an additional comparative study. Figure 6.21 shows the result of validating the best performing Viola&Jones and ACF detector trained on the top left quarter and then validated on the same remaining image as validation. This clearly shows that ACF outperforms Viola&Jones with a 7.5% higher average precision.

Deeply learned object classification

Our initial attempt at training a complete deep learning classification model from our limited set of training data did not produce usable results, although the model seems to converge. With a top1-accuracy for classification of only 33% given a two-class problem *(coconut tree or background)*, this trained model performs worse than random guessing on the class label, which given a large enough dataset, should eventually converge to 50% top1-accuracy.

1. Transfer learning with frozen layers

The transfer learning using TensorFlow is done with only 75 coconut tree samples and 75 background samples, randomly sampled from the dataset, because retraining the final convolutional layer is computationally less demanding. All other layers are frozen in this set-up, meaning their weights cannot be changed. The remaining image content is used for validation and compared to the ground truth annotations. The trained classification model achieves a top1-accuracy of 77%, a validation metric used in large-scale classification benchmarks.

To be able to compare this accuracy to the accuracy of our previously trained boosted cascades we calculate the precision and recall at pixel level. This results in a precision of 72% at a recall of 52%. Compared to the results obtained with our boosted cascades we decide that his approach does not yield satisfying results, and thus this approach was abandoned.

2. Transfer learning by fine-tuning all layers

Following the frozen-layer-model approach, we suggest using pre-trained weights as initialization for model fine-tuning. However, instead of freezing the weights of all but the last convolutional layers, we allow the complete network to fine-tune its weights.

We started with the default *Darknet19* network, existing of 19 convolutional layers and then tried a similar approach with the more complex *Densenet201* network, containing 201 convolutional layers. The reason for testing both architectures is the fact that the author of the Darknet framework illustrated that using an even deeper network achieves higher top1-accuracy while being slower at inference time [95]. We decided to verify if this behaviour was reproducible using our coconut tree dataset.

Figure 6.22 displays the loss rate versus the number of training iterations for both models. As seen both models seem to be able to converge to a stable model given enough iterations. In order to avoid over-fitting to our training data, we evaluated our deep learned classification models at several iteration intervals to determine the best model weights for our coconut tree classification task. The fast drop in loss rate is explained by the fact that we increase the batch size for training these models. This allows to take larger learning rate steps and at once is some sort of safety measurement against outliers. For our fine-tuned *Darknet19* model we find that using 10.000 iterations seems optimal at a top1-accuracy of 95.2%, while for the fine-tuned *Densenet201* model using 20.000 iterations gives us the best performance at a top1-accuracy of 97.4%.

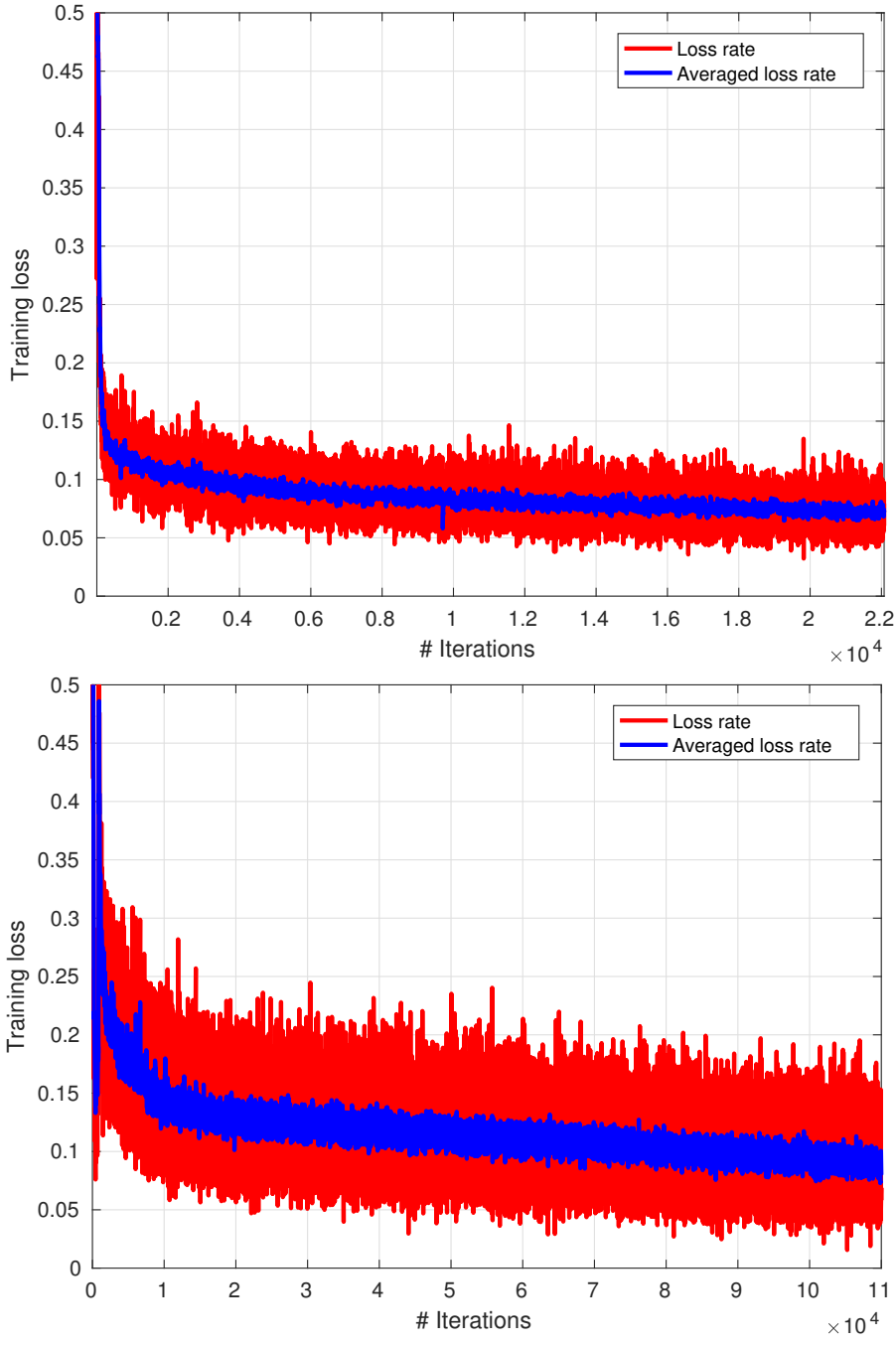


Figure 6.22: Loss-rate and average loss-rate curves during training for (top) Darknet19 and (bottom) Densenet201 model.

Table 6.4: Execution speeds for a full 10.000×10.000 pixel image for both deep learning models (*Darknet19* and *Densenet201*), at different step sizes.

step	#patches	Darknet19	Densenet201
5px	3.924.361	4h	20h30m
25px	157.609	9m5s	50m20s
50px	39.601	2m30s	12m35s

3. Execution speed and memory footprint of deep learning

Since we are using the classification architectures of Darknet instead of using detection architectures, we are aware that we need to apply a sliding window based evaluation on our large input image to perform coconut tree localisation.

We evaluate our models using a single NVIDIA TitanX GPU. The *Darknet19* model evaluates our 100×100 pixel input patches at 265 FPS while our *Densenet201* model evaluates patches at 52 FPS. Table 6.4 gives an overview of the step size in relation to the execution time of our models.

Due to the nature of our images, performing a multi-scale analysis is useless, since images are captured on a constant flight height. Given the fact that deep learning is quite robust to slight size changes, we stick to a 100×100 patch.

Considering a 50-pixel overlap between patches in both directions and thus at a 50-pixel step size, the complete image can be evaluated in only two and a half minutes using our *Darknet19* model. While this does increase when a smaller step size is selected, one can argue if this smaller step size is actually needed, since there is already a 50% overlap of patches in both dimensions. Given the fact that there are several more optimization possibilities (e.g. using multiple GPUs) makes us believe that we can achieve even faster processing speeds.

If we compare this to our boosted cascade based approaches, our Viola&Jones model takes 10 minutes for a 10.000×10.000 pixel image, while the ACF model takes 5 minutes for the same resolution. However, both models have a pixel step size of 5 pixels and thus perform in that case much faster than the deep learning frameworks. Given the high top-1 accuracy obtained with the deeply learned models, one could definitely consider switching to these more advanced algorithms, but considerations for speed optimisations should be taken.

Finally, taking a look at the memory footprint of our deep learning models might be interesting for future research. For training on our NVIDIA TitanX instance, we made the batch sizes as large as possible to fill the complete 12GB of dedicated memory. However, at runtime, we process image patch per image patch and thus the footprint is only 400MB for both models, which means the model can also be run on a low-end GPU, albeit slower.

Figure 6.23: Visual results for the **(top)** VJ boosted cascade model *[P=90.46%,R=81.12%]*, the **(middle)** ACF boosted cascade model *[P=90.55%,R=86.43%]* and the **(bottom)** deep learned Darknet19 model *[P=97.31%,R=88.58%]* showing: *(green)* TP *(red)* FP *(purple)* FN.

Visual Results

Precision-recall curves or top1-accuracy results give a quantitative evaluation of the trained models, but for customers, it is always interesting to see visual results of the trained models. Therefore we developed a visualisation tool that allows visualising the output detections of any given model with a specific colour code as an overlay on top of the original input image, as seen in Figure 6.23. Here we see the output of our VJ and ACF boosted cascade algorithms and for our deep learning classification output. For visualisation purposes, we need to select a fixed point on the precision-recall curve. In our case, we selected the most optimal configuration for each detector, as seen in Table 6.5.

The threshold is set at a precision of 90.46% and a recall of 81.12% for the Viola&Jones model, a precision of 90.55% and a recall of 86.43% for the ACF model and a precision of 97.31% and a recall of 88.58% for the deep learning approach. Green patches are true positive detections *(patches classified as coconut tree by the model and actually containing a coconut tree)*, red patches are false positives *(patches classified as coconut tree by the model but not containing a coconut tree)* and purple patches are false negatives *(patches classified as background by the model but actually containing a coconut tree)*.

Comparing the different output images, we clearly see some expected behaviour. The Viola&Jones model suffers from a higher false positive rate than the ACF model. This can be explained by the fact that VJ does not take into account colour information and thus triggers several detections on coconut tree shadows, whereas ACF is more robust to this. Comparing the ACF model to the Darknet19 model, we see that the Darknet19 model has almost no false positive detections, hence the high precision at a high recall rate. However, the approach still suffers from false negative detections. We are convinced that this is partly due to the step size of 50 pixels, used for this evaluation. Decreasing the step size towards 25 or even 10 pixels, should further reduce the number of false negative detections. It is also possible to take a look at region proposal networks, to combine with our classification deep learning networks. This would reduce the number of image patches drastically, immediately decreasing the number of false positives, ensuring an even faster pipeline.

Table 6.5: Configurations for the visual output, including precision, recall, training and inference time (for a 10.000×10.000 pixel image).

Model	Precision	Recall	Training	Inference
Viola&Jones	90.64%	81.12%	2h	10m
ACF	90.55%	86.43%	30m	5m
Darknet19	97.31%	88.58%	24h	2m30s

6.2.5 Conclusions on coconut tree detection in aerial imagery

With this research, we have proven both the capabilities of boosted cascade as well as deeply learned detection models for coconut tree localisation in aerial images. Our best boosted cascade performs at an average precision of 94.56% while our best deep learning model achieves a top1-accuracy of 97.4%. Although our deep learning pipeline evaluates two times as fast, we reckon that boosted cascades are still in the race, especially given the lower computational complexity demands, but the high classification accuracy and speed of deep learning can simply not be ignored.

Hitting a top1-accuracy of 33% on a deep model learned without initialized weights proves again that given a limited set of data training your own model without transfer learning is not feasible. We are convinced that this model is heavily over-fitted on the augmented limited training data and thus unable to generalise over a larger validation dataset. Alternatively, the training might be stuck in a sub-optimal local minima instead of a global minimum on the loss-rate surface, and unable to exit this minima without more qualitative training data.

6.3 Conclusion: benefits of using deep learning approaches for object detection

In this chapter we took a look at the possibilities of using deep learning for industrially oriented object detection, certainly given the availability of large public datasets, available pre-trained deep learned models and of course affordable hardware. Section 6.1 illustrated the use of transfer learning, where we use a pre-trained deep learned model and a small set of application-specific training data to build a new case-specific deep learned model for accurate and robust object detection. Furthermore, we demonstrated this approach on several relevant applications gaining a high average precision for each detector. In the case of the promotion board detector we did not succeed in detecting all promotion boards, nevertheless, we succeeded in achieving our goal, which is notifying if a persons has looked a specific promotion board. All boards trigger at least one detection in a specific viewpoint, making the instance based detection accuracy 100% even with an average precision that is far lower. For the cigarette box detector, we achieve a flawlessly detection and classification system, which is by the moment being integrated by our industrial partner into their augmented commercial application.

Finally, this section clearly illustrates when using deep learning, the difference between using a classification and a detection based approach can be very small and thus double pipelines can be easily integrated into a single robust solution.

Section 6.2 investigated the use of deep learning for object detection in aerial imagery, but furthermore compared the possibilities of deep learning towards boosted cascade based object detection approaches. We proved that although boosted cascades are still a valid and accurate solution for many industrial problems, that we simply cannot ignore that achieved accuracies when throwing deep learning into the battle.

Chapter 7

Lessons learned for industrial object detection tasks

The work presented in this chapter was partially published in the co-authored 'OpenCV 3 Blueprints'-book, chapter 5: 'Generic object detection for industrial applications.' [45].

After 5 years of research in the field of object detection and more specifically in industrially relevant object detection, we reckon that we have learned some critical insights in how new industrially relevant object detection tasks should be tackled. This chapter discusses the general lessons learned and gives an overview of the decision process we take whenever being faced with a new object detection challenge by an industrial partner.

7.1 Does the problem need object detection?

Since our research is demand-driven based on actual issues in the industry, we generally experience that the first task we are faced with is actually analysing if the problem statement can actually be solved by object detection. We notice that many industrial partners get lost in the difference in concepts between object classification, object recognition and object detection. It is quite important to get that difference clear from the start, in order not to develop any solutions that might prove to be conceptually broken in the long run.

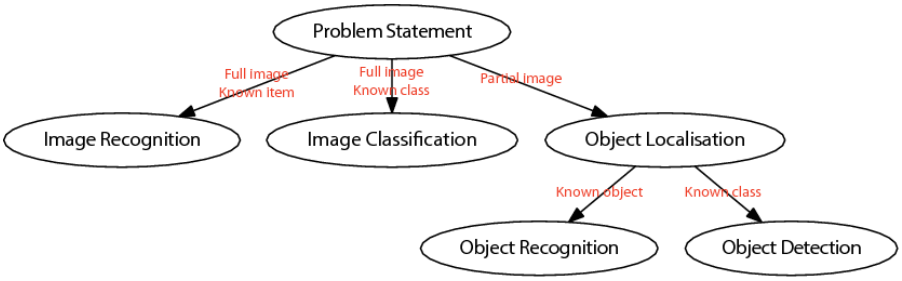


Figure 7.1: Decision tree defining the actual approach needed for the problem.

Figure 7.1 explains the decisions needed to define whether or not our proposed object detection algorithms should be used or not. We start with the actual problem statement and the input image. If one wants to label a complete image, we apply image recognition if we know the exact content of the image or we apply image classification if we know the image class, e.g. some sort of tomato. However, this will only tell you what object instance or what object class is in the entire image, but not where it is located in relation to the complete image view. If we want to specify which part of the image contains the object, we need object localization. In that case, we need to ask another question. If we look for an exactly known object *(e.g. looking for the Mona Lisa in the Louvre)* then we use object recognition algorithms. However, if we want to identify a class of objects *(including all possible variations in shape, lighting conditions, perspective transformations, colour, texture, etc.)* and accurately locate them in the given input image, then we need object detection as a solution to our problem. Making sure that this is clear from the beginning, right at the problem statement, clearly helped in finding the correct solution for many of our industrial cases.

7.2 Selecting boosted cascades or deep learning?

In this dissertation, we discussed two main approaches for solving industrially relevant object detection tasks. First, we introduced the concept of boosted cascades of weak classifiers, being dependent on one or more feature channels for training the actual object model.

Secondly, we introduced deep-learned algorithms for the same task, and more specifically the use of either a combination of sliding window and convolutional neural networks focussing on classification or a single-shot deep-learned object detector.

If a new object detection task comes up, we suggest looking at some specific elements that can point you to the actual decision of which approach to test. In the following cases, given our experience, we suggest that one should always use the more advanced deep-learned approaches.

1. If you have a large set of annotated and case-specific training data available. Training data is the key to successful deep learned approaches, even though we proved that using transfer-learning it is possible to make robust object detectors with limited data. However, having the data allows you to train complete specific models to your case that will probably push accuracy even higher than transfer-learned models can achieve.

2. If using GPGPU hardware is an option in your set-up, do so. Their costs have dropped drastically in recent years and they are worth the investment.

3. If you simply want the highest accuracy possible, without caring about the actual processing speeds. In that case, fine-grained sliding-window based approaches combined with deeply learned classification networks will probably beat every other approach.

4. If you do not need to know what your network is doing internally, or at least not every single detail. We still notice that, although all the academic investments in visualizing the internals of deep-learned networks, this is still a breaking point for many industrial partners, who are not yet convinced that we can explain why a deep-learned model is actually working.

However, as already proven in chapter 6, boosted cascades are still a valid alternative in many situations, where guaranteeing the above conditions is simply impossible. Therefore we generally consider these more classical approaches as a valid alternative in the following cases.

1. If explain-ability and understandability are key in the solution, then boosted cascades have a huge advantage over neural networks. It is perfectly possible to tell which features are exactly used to build up the weak classifiers. Furthermore, the type of features fed to the boosting process is in most cases manually crafted features that actually tell us something we can grasp as humans, as compared to the convolutional filters used by deep-learned networks.

2. If you have a very constrained scene, using those constraints can drastically reduce the complexity of the detection task *(as proven in chapter 3 and 4)*. In those situations training a boosted cascade classifier and performing inference is less complex, faster and eventually achieves similar accuracies.

3. If you have an embedded application *(e.g. a vision-navigated drone)*, then in many cases it is not feasible to add a heavy and power-consuming GPGPU. It will drastically reduce the battery lifespan for a single flight, reducing the applicational field. This is one of the main reasons why boosted cascades are still frequently used on multi-core embedded platforms.

4. Even if your mobile platform is able to add a GPGPU *(e.g. the NVIDIA Jetson TX2)* that is more power efficient, we notice that the performance speed at inference time for deep models drastically drops, to a couple of frames per seconds. In many cases, this might be enough, but there is a wide range of cases where obtaining real-time performance is mandatory *(e.g. when implementing obstacle avoidance on a high-speed drone)*. In those cases using boosted cascades is a feasible alternative.

5. If extreme execution speeds is more important than achieving top-notch accuracy *(e.g. when you need to process millions of images a day)*, then boosted cascades easily beat deep learned models, certainly when moving to high-resolution images. In those cases missing a few objects will be acceptable if we can manage to increase the number of processed images.

We reckon that in all other cases, where these guidelines do not explicitly give you an idea on what to do, a fair comparison between both techniques is needed, both in training complexity, training time, inference time. Only then one will be able to decide which algorithm actually defeats the other.

7.3 The importance of the correct training data

Selecting the correct training data is important for many computer vision tasks, but we notice that for object detection, that it is even more so. In this section, we will discuss how selecting the correct training data influences each algorithm because it slightly differs. Furthermore, we will also highlight some dangers of using techniques like data augmentation to generate more data than you have actually available.

Object detection depends on training data, it's as simple as that. And in general, when building object detectors that have to work in every single situation, adding more training data to the process results in better performing object detection models. However, this is not the case for application-specific object detection models. Due to the scene- and application-specific constraints, building a model with limited data is actually feasible, keeping in mind that this model will only achieve its maximum performance in that given application and will probably not generalize that well. However for many industrial applications, this is exactly the case and we do not focus on generic models, like in academics.

7.3.1 Training data when using boosted cascades

To gather positive and negative training samples for training boosted cascades, we suggest to follow a set of simple rules:

1. For the positive object samples only use natural occurring samples that have a relation to your application. Collecting samples in for example lab-like conditions will never result in an accurate detection model in the application itself. In our opinion, it is better to use a small set of decent high-quality object samples, rather than using a large set of low-quality non-representative samples for the application.

2. For the negative background samples there are two possible approaches, but both start from the principle that you collect negative samples in the situation where your detector is going to be used, which is very different from the normal way of training object detects, where just a large set of random samples not containing the object are being used as negatives.

   (a) Either point a camera at your scene and start grabbing random frames to sample negative windows from.

   (b) Or use your positive images to your advantage. Cut out the actual object regions and use the rest of the image content as negatives for window sampling. Keep in mind that in this case the ratio between background information and actual object occurring in the window needs to be large enough.

3. If your background is constant and not that many variations can occur, then restrain yourself trying to collect many negatives. E.g. in the case of a constant background colour, as in the case of our orchid flower detector, use only background samples of that specific background.

In some cases, this might seem contradictory to drop data in the goal of achieving better detectors, but using case-specific training data will allow you to build models that are very robust in that specific application. Furthermore, it will also allow dropping training time drastically together with a reduced amount of manual annotations needed for training these models.

Finally, we noticed that data augmentation for training boosted cascades is dangerous. Data augmentation can generate that many variations of your initial training data, leading to a model that over-fits to those artificially changed images, instead of focussing on modelling the actual object. A lot of online object detection tutorials for boosted cascades use a single image-based approach, generating many images on top of those with slight deformations. However, we notice that this only works for very simple toy samples with a simple background, but for actual industrial applications this seems to fail. E.g. in the case of trying to detect a banana on a clean white table, taking one annotated banana image and then using data augmentation to generate 100 views of that specific banana, will make a good banana detector, as long as no clutter and other objects are introduced in the system. This is mainly because the features calculated on the object pixels and background pixels are easily separable in this context. In industrial cases where clutter, illumination changes, shadows, etc. are common, this approach will trigger tons of false positive detections on the background information and give a serious drop in true positive detections.

7.3.2 Training data when using deep-learned approaches

When collecting training data for deep learning approaches, in principle the suggestions in the above subsection still hold. Even in the case of deeply learned models, it is wise to build case-specific models and use case-specific data for training. Of course, there are some differences that should be highlighted.

1. If you want to train a completely new architecture and you do not have pre-trained weights on which you can apply transfer learning, then you will have to collect a lot of manually annotated training data. Even with data-augmentation applied, you will still need multiple ten-thousands of annotated training samples to be able to converge to a good performing detection model.

2. If you base your model on an existing architecture and you have pre-trained weights on a larger more general dataset, you can easily build robust, application-specific deep learned models with as less as 50 training samples. So before annotating more, try using a small amount and see where that gets you.

3. Watch out for data augmentation when it comes to deep learning. Since deep learning applies this in various ways by default, you need to ensure which data augmentation steps your pipeline implements exactly and see if that is desired in your case. E.g. if your scene constraints allow for any kind of colour-based filtering, make sure that your deep learned detector does not apply colour transformations that generate samples outside the allowed range.

7.4 Algorithm specific parameters

In our five years of research working on this dissertation, we collected some insights and no-go's when training detection models. In this last section, we want to quickly highlight some crucial things to keep in mind when training your own detectors. Again we will discuss boosted cascades and deep-learned architectures separately.

7.4.1 Parameter tuning for boosted cascades

Boosted cascades have several parameters that can greatly affect the performance of the detection model. The summation beneath is just an overview of some parameters we experimented with and on which we have some valuable insights, given the used OpenCV 3 framework and its boosted cascade implementation.

- Your model can be quite compact. Knowing that the original Viola & Jones face detector only had a 24x24 pixel dimension, we suggest not to make it too big. The dimensions of the model immediately define the smallest possible object you are able to detect without up-scaling your image, so keep this in mind. In general, we suggest to take half of the average object dimensions in your training samples and keep your final model dimensions smaller than 100 pixels.
- We clearly noticed that LBP features, due to their binary nature, are faster both at training and inference, compared to HAAR-like wavelet features, without dropping much of accuracy. For proof-of-concept testing we thus suggest using these LBP features.
- The depth of your weak classifiers directly influences the complexity of your model and thus the inference time of your detection model. From experience, we suggest using stumps, which are simple binary decision trees, which seem to be more than capable of modelling robust and accurate cascades.

- You need to define the minimal hit rate on the positive samples for each weak classifiers. We noticed many times that user set this value too strict, resulting in a detector that is not able to converge. Whenever this happens, try lowering this value with small steps at a time.

7.4.2 Parameter tuning for deeply learned detectors

For deep-learned detection models using Darknet and the YOLOv2 architecture, we also experienced some specifics that are worth mentioning and considering when training new models.

- When using transfer learning to create case-specific object detectors we know the learning rate is critical. In order to ensure that our model does not lose its learned properties from the pre-trained weights, we suggest setting the learning rate lower than the learning rate of the pre-trained weights.
- From our limited experience, we notice that our retrieved deep learned models converge towards a stable configuration once the returned average loss-rate drops below 0.5. For new cases, we use this as a default setting to halt the training. In order to achieve the best configuration one still needs to cross-validate with a separate validation set.
- Since in transfer learning data augmentation is applied to generate thousands of samples from a very limited manually annotated training set, we stress the importance of making sure that those limited annotations are done correctly. Due too the data augmentation one or two wrong annotations can already have a big influence on the accuracy of the final detection model.

7.5 A conclusion on lessons learned

In this chapter, we produced a valuable list of learned lessons. We explained the key difference between recognition and detection, gave valuable insights in the data collection process *(which is specific for each type of algorithm)* and proposed several hints on how to choose your parameter settings wisely when using the object detection algorithms as we proposed them.

Chapter 8

Conclusion and future work

8.1 General conclusion on this dissertation

In this dissertation, we investigated how we could use scene- and application-specific constraints and knowledge, to improve the performance of state-of-the-art object detection algorithms for industrial applications. The research was motivated by the fact that academic research tries to focus on robust detection in in-the-wild conditions using expensive hardware and large datasets *(which is not feasible in industrial, application-specific object detection situations)*, while many industrial applications are constrained or have conditions where several external influences are removed due to the set-up specific conditions.

We clearly proved that scene- and application-specific constraints allow us to achieve highly accurate detection accuracies, using a very limited set of training data. Like seen in chapter 3 and 4 we do this by either using the constraints as efficient pre- or post-filtering options, or by directly integrating the constraints in the training process of the object detection models themselves. In doing so we lower the number of needed training samples drastically, allowing to continuously build new and robust object detection models for the versatile object detection tasks we solved in this dissertation.

Besides looking at the scene- and application-specific constraints we also took a look at the enormous amount of manual labelling work. By smartly integrating an active learning approach in chapter 5 we succeed in supplying only a very small amount of meaningful training samples to the human annotator, drastically reducing time and cost of the manual labelling task of training data.

We succeeded in significantly dropping the number of training samples from multiple thousands to a couple of hundreds, still achieving top-notch detection accuracies and thus making object detection feasible for robust and accurate industrial solutions.

Finally we took a look at the currently state-of-the-art object detection algorithms using deep learning in chapter 6 and conclude that we can no longer ignore this new range of powerful object detection algorithms for industrial applications. With very limited sets of only ± 50 training samples and by smartly integrating data augmentation and transfer learning, we succeed in retraining existing deep learning architectures for a case-specific object detection task.

In comparing deep learning architectures to more classical computer vision algorithms for object detection we acknowledge that deep learning can give that extra gain in accuracy, but also agree that classical approaches still achieve top-notch comparable results, many times at higher inference speeds than deep learning. This makes them still very suitable for industrial object detection solutions.

In the end, we conclude that we succeeded in proving the benefits of integrating scene- and application-specific knowledge to get robust and easy to use object detection algorithms. Furthermore we succeed in proposing several solutions for reducing the amount of manual annotation drastically while still guaranteeing high accuracies. All the time we kept real-time *(varying from 10 to 30FPS at a minimum for different applications)* processing as a hard constraint and achieved in doing so.

8.2 Future work and possible expansions

We acknowledge that not all issues are yet solved. Many of our solutions need more application specific research to further improve the gained accuracies and obtain the optimal solution to the problem.

In general, we urge that future research should perform a comparative study of adding more training data to the training process and trying to automatically define an ideal setting for a given detection task. This would allow giving an initial guess in advance of how many training samples should be used for a specific detection task, probably directly related to the complexity and the constraints of the given application.

We would also want to revisit some of the solutions we proposed in the initial chapters with scene-constraints and compare them to detectors build with deep learning based approaches, even taking it one step further and integrating all the found scene- and application-specific constraints directly into the deep learning.

In the case of our orchid flower detection system, developed in chapter 3, we want to apply a similar object detection pipeline for detecting flower buds to ensure that similar results can be achieved and to further expand the automatic analysis of orchid plants. In the same chapter, we discussed the application of detecting walking aids. Here we discovered that having enough pixel information for an object class is vital in obtaining satisfying accuracies. Therefore we suggest a study of the number of pixels per object instance compared to the obtained accuracies, to see if we can find a meaningful and scalable relation. In the case of a walking cane, we are convinced that by simply increasing the video resolution, where the cane at least covers 10-15 pixels in its smallest dimension, we could easily detect walking canes with our used approach so far.

In our dissertation, we built a specific object detection model in relation to the application. One thing that we did not investigate is the difficulties of training an object model that generalizes better within a set of similar cases. For example with the walking aid detector, building a more generic walker model that allows detecting multiple walker types by just one generic object class model, could certainly open up new solutions. Of course, in this case, we would need more training data and probably the limitation of a short training time should then be discarded.

One thing we never really found the time for during this PhD was a deep study of the parameters of our boosted cascades and deep learning algorithms, to further fine-tune our obtained results. The tremendous amount of parameters that can be selected *(e.g the complexity of the weak classifiers, the number of feature channels, ...)* allow for further investigation trying to identify key parts of the algorithms that could use improvements.

Since we only spent around six months on deep learning, we reckon this is the first place to continue our quest for the optimal solution. First of all, we should investigate why some of our deep learned object detection models yield only moderate average precisions, while others seem to sky-rocket. Localizing the exact reasons for this could give us a better insight at where to improve these object detection pipelines. We can only guess towards the actual reasons, but we are convinced that one of the issues could well be the sampling of the video material, which in several cases contains huge amounts of motion blur *(e.g. the eye-tracker experiments in chapter 6)*.

Better capturing hardware could be a possible solution here, or replacing the validation data with only clear non-blurred data, removing the actual motion blurred evaluation frames. We reckon our current models will never detect motion blurred objects probably since we never used them as actual training data. Adding them to the training data set might improve the generalization capabilities of the deep-learned models.

A challenge in deep learning still lies in expanding the multi-class models. For now, our model still needs an overnight training step. For most applications this is feasible, but there are still applications where this is too long. We should thus investigate how we can optimally expand existing models with an extra class, at a minimal processing cost. This research field is called incremental learning and already has several application domains, like object detection in video sequences, as described by Kuznetsova et al. [56].

Finally, quite lately in our research, we discovered that data augmentation can actually worsen the performance of the detection model if it applies transformations that generate training samples that break our scene- and application-specific conditions. E.g. when you are integrating a specific colour-based filtering like suggested in section 4.1, we do not want our data augmentation to apply colour-based transformations, as suggested in section 6.1.3, because that would actually break our application-based constraint. We, therefore, suggest carefully evaluating each technique for performing data augmentation and consider if these augmentations actually make sense for any specific application.

We acknowledge that the holy grail for object detection in industrial applications would be a one-shot-learning approach, as suggested by Fei-Fei et al. [30]. This would allow us to quickly build several robust solutions for industrial applications. However reported accuracies on these techniques are, for now, still quite low and thus still not usable in our specific contexts. Also taking a look at object discovery approaches like the technique proposed by Abbeloos et al. in [3] could be a logical next step in this research.

Chapter 9

Valorisation

In this chapter, we focus on the industrial valorisation of this dissertation. Since the research is performed at EAVISE, a research group that tries to close the gap between academics and the industry, much of the research was directly returned as valuable content for the cooperating companies. This means that the valorisation of the research discussed in this dissertation is happening right now. Most of the developed software is demand-driven, starting from an actual issue within an industrial application, that we then tackle using computer vision and artificial intelligence algorithms.

We discuss several applications that are inspired by the performed doctoral research, as discussed in previous chapters, and which made it to actual implemented industrial realisations. Aside from the successful technology transfers from academics to the industry *(which basically happened for each publication and thus each specific problem we solved)*, we also touch upon the small consultancy projects in computer vision and machine learning, which are related to the PhD research work done. These are most of the time small side-projects, assigned by our supervisor and based on the capacities of the PhD student, directly using the obtained experience and knowledge from the doctoral research, to quickly turn small issues into solved problems.

Finally, we organised several workshops and symposia to share the gained knowledge among industrial and academic partners and provided several publicly available datasets, in order to allow other researchers and industrial partners to reproduce the achieved results and mimic them on their own data.

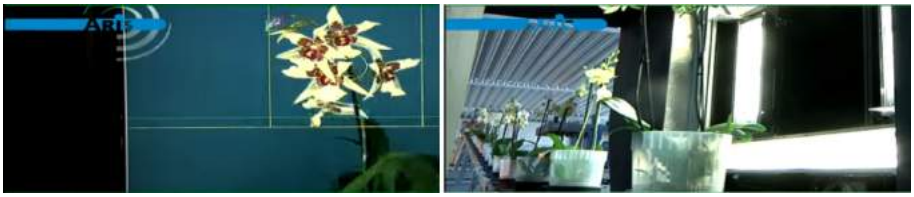


Figure 9.1: Phaleonopsis sorting and grading system by Aris BV.

9.1 Implemented industrial realisations

9.1.1 Automated orchid detection and classification system

In chapter 3, section 3.1, we discuss how we developed a fully automated pipeline for orchid detection and classification into rough texture-based classes. Combined with an analysis of the colour histogram of the flower, this provided enough information to create a fully automated species classification pipeline. This research, performed for Aris BV, was further developed inside the company and integrated into their automated orchid analysis and sorting system. Furthermore, the system is since then actively deployed on several locations in The Netherlands, as seen in Figure 9.1.

9.1.2 Strawberry picking robot

In chapter 4, section 4.1, we discuss an effective computer vision pipeline for detecting and classifying ripe strawberries. The developed software and the used approaches for cluster separation were combined with approaches developed by other research institutes, in order to obtain a full robotic strawberry picking

Figure 9.2: Strawberry picking robot by Octinion *(version 2017)*.

robot, as seen in Figure 9.2. This robot is still under heavy development, but a first prototype was introduced to the public in 2016. This robot allows farmers to guarantee production, using robots to harvest strawberries during the night and use manual labour to retrieve the remaining strawberries during the day. This will optimize cost and production and help to cope with the issue of finding personnel.

In September 2017, the first complete version of the strawberry picking robot was presented at the International Strawberry Congress in Antwerpen *(Belgium)*, organized by Hoogstraten, a co-operative company that offers healthy, safe and sustainably produced food.

9.2 Consultancy projects

9.2.1 KNFB Reader: helping blind people read

Together with Sensotec, a Belgian company developing hardware, software and applications for the (nearly-)blind and people with reading disorders, we worked on an application called the KNFB Reader. This application allows (nearly-)blind people to point their mobile phones towards a piece of text, which is automatically located and parsed with an optical character recognition (OCR) interface, speaking the text out loud. This helps disabled people to interact better with their surroundings.

Several computer vision related sub-tasks were performed during this consultancy:

1. I developed an algorithm that localizes and classifies *(does the page contain actual printed content)* printed pages.
2. I generated software that automatically takes a picture from the moment no movement is detected, which can be used in combination with a cardboard scanning device stand, for scanning multiple pages.
3. I developed software for defining page orientation and correction.
4. I developed software for efficient text segmentation and de-skewing using the stroke width transform approach.

The application has won several prices and multiple Golden Apple awards. It is available for iOS, Android and Windows10. It has been downloaded over 100.000 times so far aiding visually impaired people all over the world.

9.2.2 Optidrive: automation through computer vision

For a Belgian robotics company, Optidrive, we developed several proof-of-concept software solutions for robotics and computer vision related problems they wanted to see solved.

1. I developed an algorithm that does a surface analysis of large marked stone panels, in order to guide a sanding robot towards the correct location.
2. I developed a pipeline for concrete drilling core analysis, looking for an automated way of separating and classifying the actual layers.
3. I developed a highly accurate doughnut localization algorithm, needed for robotic arm positioning on plastic plates before processing them.

The core idea of this consultancy task was trying to maintain real-time performance in combination with sub-millimetre accuracy in the localization process. Besides doing the algorithm development, we also looked into usable camera hardware.

9.2.3 OneUp: automatic ticket analysis

For a small Belgian start-up company, OneUp, we did some OpenCV based computer vision consultancy. The task existed of robustly localizing, and analysing tickets from a variety of consumer shops, using a mobile phone application, while coping with challenging scene- and application-specific conditions like different orientations, sizes, illuminations, ...

9.3 Workshops and symposia

In order to optimally spread the developed research output, I organized several hands-on workshops and symposia, where I tried to disperse my experience towards industrial partners. At each event, industrial partners were allowed to bring their own specific computer vision related problems to discuss possible solutions.

9.3.1 Hands-on workshops for industrial object detection

In the context of the IWT-TETRA project TOBCAT *Industrial applications of object categorization*, I organized a workshop on object detection techniques using the OpenCV embedded Viola and Jones framework for generic industrial object detection. Three workshops were organized during the period of February and March 2014, in order to make industrial partners aware of the possibilities of object detection techniques. The workshops, accompanying place to 40 participants, were a perfect place for bringing theory and practical examples together forming concrete solutions for industrial related problems.

9.3.2 First edition of the AAA Vision symposium

The first edition of the Symposium on industrial Applications of Advanced computer vision Algorithms, organized in September 2014, was the closing symposium of the IWT TETRA project TOBCAT discussing the project end-results in detail. The symposium, with 150 participants, was combined with a computer vision based demo fair containing more than 25 vision based demo set-ups, from both academic and industrial partners. More info on this symposium can be found at `http://www.eavise.be/AAAVision/`.

9.3.3 Deep learning workshop

When focussing the final months of our doctoral research on deep learning, we discovered that several industrial partners showed great interest in the possibilities of deep learning. Therefore I organized an introductory workshop on deep learning, covering the general principles of deep learning and the possible applications, especially related to the field of my research, being object detection. The four-hour-long workshop was requested by Aris BV, a company from the Netherlands, and was held for 25 participants in May 2017.

Figure 9.3: Logo of the first edition of the AAA Vision Symposium.

9.4 Public datasets

During the PhD research, several datasets were collected, annotated and made publicly available in order to allow people to reproduce our obtained results. Below you can find an overview of those datasets.

- EAVISE Mobile Mapping Dataset
  - The EAVISE Mobile Mapping Dataset consists of two annotated mobile mapping datasets, existing of mobile mapping cycloramic images, captured using a LadyBug 1 and LadyBug 2 camera set-up. The first set is captured in a quiet and calm urban area in the Netherlands and contains 450 images under daylight conditions with a resolution of 4.800×2.400 pixels. The second dataset is captured in a train and bus station area in Belgium and contains 45 images with a resolution of 8.000×4.000 pixels.
  - Found at: `http://eavise.be/MobileMappingDataset/`
- EAVISE Solar Panel Dataset
  - The EAVISE Solar Panel Dataset contains 2.500 solar panel annotations and a larger image 16000x16000 aerial image with solar panels blacked out, containing the 150.000 negatives random samples collected for model training. We provide a test image of a single 2×2km urban area (Sint-Truiden, Belgium) with a resolution of 25cm/pixel combined with solar panel installation annotations. The image has an original resolution of 8000×8000 pixels but was up-scaled using bi-cubic interpolation to a resolution of 16.000×16.000 pixels to provide enough pixel information per individual solar panel (19×14 pixels).
  - Found at: `http://www.eavise.be/SolarPanelDataset/`
- EAVISE Open Source Face Detection Dataset
  - EAVISE Open Source Face Detection Dataset consists of several items that were used to generate the improved frontal face detection model using LBP features and AdaBoost for OpenCV3.2. It contains the annotation of the FDDB dataset, converted to the correct OpenCV format. Furthermore, it contains the final trained model which is included in OpenCV3.2.
  - Found at: `http://www.eavise.be/OpenSourceFaceDetection/`

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[3] W. Abbeloos, E. Ataer-Cansizoglu, S. Caccamo, Y. Taguchi, and Y. Domae. 3d object discovery and modeling using single RGB-D images containing multiple object instances. In *Proceedings of the IEEE International Conference on 3D Vision (3DV)*, 2017.

[4] T. Ahonen, A. Hadid, and M. Pietikäinen. Face recognition with local binary patterns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 469–481, 2004.

[5] G. Baldewijns, G. Debard, T. Croonenborghs, and B. Vanrumste. Semi-automated video-based in-home fall risk assessment. In *Proceedings of the Association for the Advanced of Assistive Technology in Europe Congress (AAATE)*, pages 59–64, 2013.

[6] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Pedestrian detection at 100 frames per second. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2903–2910, 2012.

[7] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool. Seeking the strongest rigid detector. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[8] Y. Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of the International Conference on Machine Learning (ICML), Unsupervised and Transfer Learning Workshop*, pages 17–36, 2012.

[9] T. L. Berg, A. C. Berg, J. Edwards, and D. Forsyth. Who's in the picture. *Advances in Neural Information Processing Systems (NIPS)*, 17:137–144, 2005.

[10] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems (NIPS)*, pages 523–531, 2016.

[11] A. Bleau and L. J. Leon. Watershed-based segmentation and region merging. *Computer Vision and Image Understanding (CVIU)*, 77(3):317–370, 2000.

[12] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[13] L. Čehovin. Trax: The visual tracking exchange protocol and library. *Neurocomputing*, 2017.

[14] H. Cho, P. E. Rybski, A. Bar-Hillel, and W. Zhang. Real-time pedestrian detection with deformable part models. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IVS)*, pages 1035–1042, 2012.

[15] C. Cortes and V. Vapnik. Support vector machine. *Machine Learning*, 20(3):273–297, 1995.

[16] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE, 2005.

[17] G. M. Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, P. Vincent, et al. Unsupervised and transfer learning challenge: a deep learning approach. In *Proceedings of the International Conference on Machine Learning (ICML), Unsupervised and Transfer Learning Workshop*, pages 97–110, 2012.

[18] F. De Smedt and T. Goedemé. Open framework for combined pedestrian detection. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, volume 2, pages 551–559, 2015.

[19] F. De Smedt, D. Hulens, and T. Goedemé. On-board real-time tracking of pedestrians on a UAV. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2015.

[20] F. De Smedt, L. Struyf, S. Beckers, J. Vennekens, G. De Samblanx, and T. Goedemé. Is the game worth the candle? Evaluation of OpenCL for object detection algorithm optimization. In *Proceedings of the International Joint Conference on Pervasive and Embedded Computing and Communication Systems (PECCS)*, pages 284–291, 2012.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[22] E. Dibra, J. Maye, O. Diamanti, R. Siegwart, and P. Beardsley. Extending the performance of human classifiers using a viewpoint specific approach. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 765–772, 2015.

[23] P. Dollár. Piotr's Computer Vision Matlab Toolbox (PMT). `https://github.com/pdollar/toolbox`, 2005.

[24] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(8):1532–1545, 2014.

[25] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. In *Proceedings of British Machine Vision Conference (BMVC)*, volume 2. Citeseer, 2010.

[26] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *Proceedings of British Machine Vision Conference (BMVC)*, volume 2, page 5, 2009.

[27] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features – addendum. 2010.

[28] S. R. Dubey and A. S. Jalal. Detection and classification of apple fruit diseases using complete local binary patterns. In *Proceedings of the IEEE International Conference on Computer and Communication Technology (ICCCT)*, pages 346–351, 2012.

[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[30] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(4):594–611, 2006.

[31] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[32] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2241–2248, 2010.

[33] G. Feng, C. Qixin, and N. Masateru. Fruit detachment and classification method for strawberry harvesting robot. *International Journal of Advanced Robotic Systems*, 5(1):41–48, 2008.

[34] D. Frejlichowski, K. Gościewska, P. Forczmański, A. Nowosielski, and R. Hofman. Applying image features and AdaBoost classification for vehicle detection in the SM4Public system. *Image Processing and Communications Challenges*, pages 81–88, 2016.

[35] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[36] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. *Decision Forests for Computer Vision and Medical Image Analysis*, pages 143–157, 2013.

[37] Z. Gan, R. Henao, D. Carlson, and L. Carin. Learning deep sigmoid belief networks with data augmentation. In *Artificial Intelligence and Statistics*, pages 268–276, 2015.

[38] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[39] J. Gleason, A. V. Nefian, X. Bouyssounousse, T. Fong, and G. Bebis. Vehicle detection from aerial imagery. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2065–2070, 2011.

[40] N. Gourier, D. Hall, and J. L. Crowley. Estimating face orientation from robust detection of salient facial features. In *Proceedings of the*

International Workshop on Visual Observation of Deictic Gestures (ICPR), 2004.

[41] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 345–360, 2014.

[42] S. Hagler, D. Austin, T. L. Hayes, J. Kaye, and M. Pavel. Unobtrusive and ubiquitous in-home monitoring: a methodology for continuous assessment of gait velocity. *IEEE Transactions on Biomedical Engineering*, 57(4):813–820, 2010.

[43] M. Heikkila, M. Pietikainen, and C. Schmid. Description of interest regions with local binary patterns. *Pattern Recognition*, 42(3):425–436, 2009.

[44] S. Hinz and A. Baumgartner. Automatic extraction of urban road networks from multi-view aerial imagery. *Journal of Photogrammetry and Remote Sensing*, 58(1):83–98, 2003.

[45] J. Howse, S. Puttemans, Q. Hua, and U. Sinha. *OpenCV 3 Blueprints: Expand your knowledge of computer vision by building amazing projects with OpenCV3*. Packt Publishing Ltd, 2015.

[46] D. Hulens, K. Van Beeck, and T. Goedemé. Fast and accurate face orientation measurement in low-resolution images on embedded hardware. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, volume 4, pages 538–544, 2016.

[47] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

[48] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[49] V. Jain and E. Learned-Miller. FDDB: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[50] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678, 2014.

[51] H. Jiang and E. Learned-Miller. Face detection with the faster R-CNN. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, pages 650–657, 2017.

[52] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with Gaussian processes for object categorization. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[53] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[54] S. A. B. R. W. Kressig. Laboratory review: the role of gait analysis in seniors mobility and fall prevention. *Gerontology*, 2010.

[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[56] A. Kuznetsova, S. Ju Hwang, B. Rosenhahn, and L. Sigal. Expanding object detector's horizon: incremental learning framework for object detection in videos. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 28–36, 2015.

[57] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(3):453–465, 2014.

[58] E. Learned-Miller, G. B. Huang, A. RoyChowdhury, H. Li, and G. Hua. Labeled faces in the wild: A survey. *Advances in Face Detection and Facial Image Analysis (FDA*, pages 189–248, 2016.

[59] C. M. Lee, K. E. Schroder, and E. J. Seibel. Efficient image segmentation of walking hazards using IR illumination in wearable low vision aids. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 0127–0127, 2002.

[60] W.-C. Li and D.-M. Tsai. Wavelet-based defect detection in solar wafer images with inhomogeneous texture. *Pattern Recognition*, 45(2):742–756, 2012.

[61] X. Li and Y. Guo. Adaptive active learning for image classification. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 859–866, 2013.

[62] Y. Li, B. Sun, T. Wu, and Y. Wang. Face detection with end-to-end integration of a ConvNet and a 3D model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 420–436, 2016.

[63] Y. Li, B. Sun, T. Wu, Y. Wang, and W. Gao. Face detection with end-to-end integration of a convnet and a 3d model. *arXiv preprint arXiv:1606.00850*, 2016.

[64] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Z. Li. Learning multi-scale block local binary patterns for face recognition. *Advances in Biometrics*, pages 828–837, 2007.

[65] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. In *Proceeding of the IEEE International Conference on Image Processing (ICIP)*, volume 1, pages I–I, 2002.

[66] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[67] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

[68] D. D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97, pages 211–218, 1997.

[69] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[70] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, pages 720–735, 2014.

[71] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool. Traffic sign recognition: How far are we from the solution? In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2013.

[72] H. Mayer. Automatic object extraction from aerial imagery—a survey focusing on buildings. *Computer Vision and Image Understanding (CVIU)*, 74(2):138–149, 1999.

[73] K. Milisen, E. Detroch, K. Bellens, T. Braes, K. Dierickx, W. Smeulders, S. Teughels, E. Dejaeger, S. Boonen, and W. Pelemans. Falls among community-dwelling elderly: a pilot study of prevalence, circumstances and consequences in Flanders. *Tijdschrift voor Gerontologie en Geriatrie*, 35(1):15–20, 2 2004.

[74] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[75] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[76] Y. Nakashima, T. Koyama, N. Yokoya, and N. Babaguchi. Facial expression preserving privacy protection using image melding. In *Proceedings of the IEEE International Congress on Mathematical Education (ICME)*, pages 1–6, 2015.

[77] M.-E. Nilsback and A. Zisserman. Delving deeper into the whorl of flower segmentation. *Image and Vision Computing*, 2009.

[78] Y.-I. Ohta, T. Kanade, and T. Sakai. Color information for region segmentation. *Computer Graphics and Image Processing*, 13(3):222–241, 1980.

[79] H. Okamoto and W. S. Lee. Green citrus detection using hyperspectral imaging. *Computers and Electronics in Agriculture*, 66(2):201–208, 2009.

[80] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[81] L. Paletta and A. Pinz. Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, 31(1):71–86, 2000.

[82] I. Panagiotis. Preventing privacy leakage from photos in social networks. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2015.

[83] J. Peng and B. Bhanu. Closed-loop object recognition using reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(2):139–154, 1998.

[84] P. Peng, Y. Tian, Y. Wang, J. Li, and T. Huang. Robust multiple cameras pedestrian detection with multi-view bayesian network. *Pattern Recognition*, 48(5):1760–1772, 2015.

[85] S. Puttemans, G. Baldewijns, T. Croonenborghs, B. Vanrumste, and T. Goedemé. Automated walking aid detector based on indoor video recordings. In *Proceedings of the International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5040–5045, 2015.

[86] S. Puttemans, T. Callemein, and T. Goedemé. Building robust industrial applicable object detection models using transfer learning and single pass deep learning architectures. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, 2018.

[87] S. Puttemans, E. Can, and T. Goedemé. Improving open source face detection by combining an adapted cascade classification pipeline and active learning. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, number submitted, 2017.

[88] S. Puttemans and T. Goedemé. How to exploit scene constraints to improve object categorization algorithms for industrial applications? In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, volume 1, pages 827–830, 2013.

[89] S. Puttemans and T. Goedemé. Visual detection and species classification of orchid flowers. In *Proceedings of the IAPR International Conference of Machine Vision Applications (MVA)*, pages 505–509, 2015.

[90] S. Puttemans, T. Goedemé, and S. Vanwolputte. Safeguarding privacy by reliable automatic blurring of faces in mobile mapping images. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 3, pages 408–417, 2016.

[91] S. Puttemans, W. Ranst, and T. Goedemé. Detection of photovoltaic installations in RGB aerial imaging: a comparative study. In *Proceedings of the GEOBIA Conference*, 2016.

[92] S. Puttemans, K. Van Beeck, and T. Goedemé. Comparing boosted cascades to deep learning architectures for fast and robust coconut tree detection in aerial images. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, 2018.

[93] S. Puttemans, Y. Vanbrabant, L. Tits, and T. Goedemé. Automated visual fruit detection for harvest estimation and robotic harvesting. In *Proceedings of the IEEE International Conference on Image Processing Theory Tools and Applications (IPTA)*, pages 1–6, 2016.

[94] B. Reagen, R. Adolf, P. Whatmough, G.-Y. Wei, and D. Brooks. Deep learning for computer architects. *Synthesis Lectures on Computer Architecture*, 12(4):1–123, 2017.

[95] J. Redmon. Darknet: Open source neural networks in C. `http://pjreddie.com/darknet/`, 2013–2016.

[96] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[97] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[98] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[99] L. Reyzin and R. E. Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 753–760, 2006.

[100] P. Sánchez-Friera, M. Piliougine, J. Pelaez, J. Carretero, and M. Sidrach de Cardona. Analysis of degradation mechanisms of crystalline silicon PV modules after 12 years of operation in southern Europe. *Progress in Photovoltaics: Research and Applications*, 19(6):658–666, 2011.

[101] P.-A. Savalle, S. Tsogkas, G. Papandreou, and I. Kokkinos. Deformable part models with CNN features. In *Proceedings of the European Conference on Computer Vision (ECCV), Parts and Attributes Workshop (PAW)*, 2014.

[102] A. R. Sfar, N. Boujemaa, and D. Geman. Vantage feature frames for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 835–842. IEEE, 2013.

[103] F. Shaikh, A. Sharma, P. Gupta, and D. Khan. A driver drowsiness detection system using cascaded AdaBoost. *Imperial Journal of Interdisciplinary Research*, 2(5), 2016.

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

[105] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems (NIPS)*, pages 935–943, 2013.

[106] Y. Song, C. Glasbey, G. Horgan, G. Polder, J. Dieleman, and G. Van der Heijden. Automatic fruit recognition and counting from multiple images. *Biosystems Engineering*, 118:203–215, 2014.

[107] D. Stajnko and Z. Čmelik. Modelling of apple fruit growth by application of image analysis. *Agriculturae Conspectus Scientificus*, 70(2):59–64, 2005.

[108] D. Stajnko, M. Lakota, and M. Hočevar. Estimation of number and diameter of apple fruits in an orchard during the growing season by thermal imaging. *Computers and Electronics in Agriculture*, 42(1):31–42, 2004.

[109] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT Press Cambridge, 1998.

[110] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[111] Y. Tanaka, A. Kodate, Y. Ichifuji, and N. Sonehara. Relationship between willingness to share photos and preferred level of photo blurring for privacy protection. In *Proceedings of the ASE International Conference on BigData and SocialInformatics*, page 33. ACM, 2015.

[112] N. Tijtgat, W. Van Ranst, B. Volckaert, , G. Toon, and F. De Turck. Uav embedded real-time object detection and warning system. In *Proceedings of the International Conference on Computer Vision (ICCV), UAVISION Workshop*, 2017.

[113] M. E. Tinetti. Preventing falls in elderly persons. *New England Journal of Medicine*, 348(1):42–49, 2003.

[114] S. Tokui, K. Oono, S. Hido, and J. Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS), Machine Learning Systems Workshop (LearningSys)*, volume 5, 2015.

[115] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1521–1528, 2011.

[116] D.-M. Tsai, S.-C. Wu, and W.-Y. Chiu. Defect detection in solar modules using ica basis images. *IEEE Transactions on Industrial Informatics*, 9(1):122–131, 2013.

[117] K. Van Beeck, T. Goedemé, and T. Tuytelaars. A warping window approach to real-time vision-based pedestrian detection in a truck's blind spot zone. In *Proceedings of the IEEE International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 2, pages 561–568, 2012.

[118] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the ACM International Conference on Multimedia*, pages 689–692, 2015.

[119] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I–511, 2001.

[120] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511, 2001.

[121] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–534, 2011.

[122] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Convolutional channel features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 82–90, 2015.

[123] L. Yang, J. Dickinson, Q. Wu, and S. Lang. A fruit recognition method for automatic harvesting. In *Proceedings of the IEEE International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pages 152–157, 2007.

[124] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*, pages 3320–3328, 2014.

[125] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[126] Y. Zheng, C. Yang, A. Merkulov, and M. Bandari. Early breast cancer detection with digital mammograms using HAAR-like features and AdaBoost algorithm. In *Proceedings of Sensing and Analysis Technologies for Biomedical and Cognitive Applications*, 2016.

Curriculum Vitae

Steven Puttemans was born on the 18th of October 1985 in Brugge, Belgium. He obtained a bachelor degree in Industrial Sciences - Electronics-ICT at the *Katholieke Hogeschool Brugge-Oostende* (now a KU Leuven Technology Campus) in 2009. He continued his studies at the KHBO and obtained a master degree in Industrial Sciences - Electronics-ICT in 2011. Finally, he obtained a master degree in Artificial Intelligence at the *Katholieke Universiteit Leuven* in 2012. His last master thesis, titled *Development of a validation pipeline for craniofacial reconstruction* brought him into contact with computer vision and machine learning for the first time. Triggered by this intriguing field of work, he decided to start as a researcher on a two-year research project (called *TOBCAT*) at campus De Nayer, under the supervision of Prof. Toon Goedemé, at the EAVISE Research Group. The project focussed on bringing state-of-the-art object detection techniques from academic development to real-time implementation in industrial applications. After a half year of research experience, Steven decided to start his doctoral research in 2013 under the supervision of Prof. Toon Goedemé. In this work towards a PhD degree, he focussed on applying scene constraints in combination with object detection techniques, to develop robust and accurate industrial applicable object detection algorithms. Besides doing his doctoral research, Steven is also an active participant in the OpenCV community, being one of the main forum moderators and a frequent supplier of bug-fixes and improvements of the framework.

List of publications

International conference publications

[1] S. Puttemans and T. Goedemé. How to exploit scene constraints to improve object categorization algorithms for industrial applications? In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, Barcelona, Spain, 2013.

[2] S. Puttemans and T. Goedemé. Optimal object categorization under application specific conditions. In *Proceedings of the Doctoral Consortium of the International Conference on Computer Vision Theory and Applications (DCVISIGRAPP)*, Lisbon, Portugal, 2014.

[3] S. Puttemans and T. Goedemé. Visual detection and species classification of orchid flowers. In *Proceedings of the International Conference on Machine Vision Applications (MVA)*, Tokyo, Japan, 2015.

[4] S. Puttemans, G. Baldewijns, T. Croonenborghs, B. Vanrumste, and T. Goedemé. Automated walking aid detector based on indoor video recordings. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Milan, Italy, 2015.

[5] S. Puttemans, Y. Vanbrabant, L. Tits, and T. Goedemé. Automated visual fruit detection for harvest estimation and robotic harvesting. In *Proceedings of the International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Oulu, Finland, 2016.

[6] S. Puttemans, W. Vanranst, and T. Goedemé. Detection of photovoltaic installations in RGB aerial imaging: a comparative study. In *Proceedings of the GEOBIA Conference: Solutions and Synergies*, Enschede, The Netherlands, 2016.

[7] S. Puttemans, S. Vanwolputte, and T. Goedemé. Safeguarding privacy by reliable automatic blurring of faces in mobile mapping images. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, Rome, Italy, 2016.

[8] S. Puttemans, C. Ergun, and T. Goedemé. Improving open source face detection by combining an adapted cascade classification pipeline and active learning. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, Porto, Portugal, 2017.

[9] S. Puttemans, T. Callemein, and T. Goedemé. Building robust industrial applicable object detection models using transfer learning and single pass deep learning architectures. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, Funchal, Madeira-Portugal, 2018.

[10] S. Puttemans, K. Vanbeeck, and T. Goedemé. Comparing boosted cascades to deep learning architectures for fast and robust coconut tree detection in aerial images. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, Funchal, Madeira-Portugal, 2018.

Book chapter publications

[1] S. Puttemans, J. Howse, Q. Hua, and U. Sinha. Generic object detection for industrial applications. In *OpenCV3 Blueprints: Expand your knowledge of computer vision by building amazing projects with OpenCV3*, Blueprints, chapter 5, pages 181–234. PACKT Publishing, 2015.

[2] S. Puttemans, J. Howse, Q. Hua, and U. Sinha. Efficient person identification using biometric properties. In *OpenCV3 Blueprints: Expand your knowledge of computer vision by building amazing projects with OpenCV3*, Blueprints, chapter 6, pages 235–277. PACKT Publishing, 2015.

Journal publications

[1] S. Puttemans and T. Goedemé. Optimal object categorization under application specific conditions. *Journal of Communications and Computer*, 10:1484–1496, 2013.

Professional journal publications

[1] S. Puttemans and T. Goedemé. TOBCAT: Industrial applications of object categorization techniques. *DSP Valley Newsletter*, 3:9–12, 2013.

[2] S. Puttemans and T. Goedemé. TOBCAT project results: Applying object categorisation techniques in real-life industrial cases. *DSP Valley Newsletter*, 15:9–9, 2014.

Abstract publications

[1] S. Puttemans and T. Goedemé. Optimal object categorization under application specific conditions. In *Proceedings of the European Conference on The Use of Modern Information and Communication Technologies (ECUMICT)*, Ghent, Belgium, 2014.

[2] S. Puttemans and T. Goedemé. Optimal object categorization under application specific conditions. In *Proceedings of the Research Day FIIW-ESAT/CW*, Leuven, Belgium, 2015.

FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING
EMBEDDED AND ARTIFICIALLY INTELLIGENT VISION ENGINEERING (EAVISE)
Jan Pieter De Nayerlaan 5
B-2860 Sint-Katelijne-Waver
steven.puttemans@kuleuven.be
http://www.eavise.be