FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER’S THESIS

Title:	People detection, tracking and biometric data extraction using a single camera for retail usage
Student:	Bc. Lukáš Brchl
Supervisor:	doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	Until the end of summer semester 2019/20

Instructions

The work aims to use computer vision techniques to create a framework that could analyze video sequences to extract data about people. The proposed solution must be robust regarding people detection and tracking and to obtain soft biometric statistics. The deep-learning approaches might be used. The output of the framework analysis should be data of persons’ trajectories. The data could be visualized to provide an overall understanding of persons’ behavior, e.g., to optimize marketing activities of the retail store.

1) Familiarize yourself with the task of people detection, tracking, and extraction of human characteristics (soft biometrics) using a single RGB camera in the retail environment.
2) Propose methods and algorithms that can efficiently solve the task mentioned above.
3) Implement the proposed methods and algorithms using appropriate programming language and tools.
4) Verify implemented methods on real data, evaluate their accuracy and suggest further improvements.

References

Will be provided by the supervisor.

Ing. Karel Klouda, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 22, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master’s thesis

People detection, tracking and biometric data extraction using a single camera for retail usage

Bc. Lukáš Brchl

Department of Applied Mathematics
Supervisor: doc. RNDr. Ing. Marcel Jiřina, Ph.D.

May 9, 2019

Acknowledgements

First of all, I would like to thank my supervisor Assoc. Prof. Marcel Jiřina who gave rise to this assignment but also hired me to ImproLab, which was a unique opportunity to gain new professional knowledge. I have already spent more than two great years in this group doing work I enjoyed. This opportunity has dramatically shifted my thinking further.

I also thank all my colleagues from ImproLab for creating a pleasant everyday working environment. Our regular team-building events had extremely positive repercussions on my work. Specifically thanks to Ing. Jakub Novák for proofreading.

I want to express my gratitude to my co-supervisor Prof. Kai-Lung Hua from Taiwan Tech, thanks to which I was able to work on this thesis in such a beautiful country like Taiwan. I am also grateful for his course Deep Learning for Computer Vision Applications where I could revive my current knowledge of this field. The stay in Taiwan has been a fantastic experience, beneficial for my learning and my personal growth. I am also entirely determined to return to this lovely country in the future because of the kind and helpful people that can be found here.

Many thanks to the wonderful research community that exists in machine learning and computer vision field. Without so many materials, frameworks, libraries, and open-source models, this work could not arise.

I would also like to thank my family because, despite the challenging situations I faced during the studies, they have always supported my ideas and choices and provided me with lots of love.

Last but not least, big thanks to my girlfriend Michaela, who has been tirelessly supporting me throughout the last four years and also making sure that I do not run out of food during the writing of this work.

Declaration

In Prague on May 9, 2019 ....

Citation of this thesis

Brchl, Lukáš. *People detection, tracking and biometric data extraction using a single camera for retail usage.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019. Also available from: ⟨`https://github.com/lukasbrchl/People-detection-tracking-and-biometrics-extraction-using-a-single-camera-text`⟩.

Abstrakt

Tato práce se zabývá návrhem frameworku, který slouží k analýze video sekvencí z RGB kamery. Framework využívá technik sledování osob a následné extrakce biometrických dat. Biometrická data jsou sbírána za účelem využití v malobochodním prostředí. Navržený framework lze rozdělit do třech menších komponent, tj. detektor osob, sledovač osob a extraktor biometrických dat. Navržený detektor osob využívá různé architektury sítí hlubokého učení k určení polohy osob. Řešení pro sledování osob se řídí známým postupem „online tracking-by-detection" a je navrženo tak, aby bylo robustní vůči zalidněným scénám. Toho je dosaženo začleněním dvou metrik týkající se vzhledu a stavu objektu v asociační fázi. Kromě výpočtu těchto deskriptorů, jsme schopni získat další informace o jednotlivcích jako je věk, pohlaví, emoce, výška a trajektorie. Návržené řešení je ověřeno na datasetu, který je vytvořen speciálně pro tuto úlohu.

Klíčová slova počítačové vidění, detekce osob, sledování osob, extrakce biometrických údajů

Abstract

This thesis proposes a framework that analyzes video sequences from a single RGB camera by extracting useful soft-biometric data about tracked people. The aim is to focus on data that could be utilized in a retail environment. The designed framework can be broken down into the smaller components, i.e., people detector, people tracker, and soft-biometrics extractor. The people detector employs various deep learning architectures that estimate bounding boxes of individuals. The tracking solution follows the well-known online tracking-by-detection approach, while the proposed solution is built to be robust regarding the crowded scenes by incorporating appearance and state features in the matching phase. Apart from calculating appearance descriptors only for matching, we extract additional information of each person in the form of age, gender, emotion, height, and trajectory when possible. The whole framework is validated against the dataset which was created for this propose.

Keywords computer vision, people detection, people tracking, biometrics extraction

Contents

Introduction **1**
Motivation . . . 2
Challenges . . . 4
Objectives . . . 5
Assumptions . . . 6
Thesis structure . . . 6

1 Theoretical background **7**
1.1 Relevant areas . . . 7
1.1.1 Artificial intelligence . . . 7
1.1.2 Machine learning . . . 8
1.1.2.1 Supervised learning . . . 9
1.1.2.2 Unsupervised learning . . . 9
1.1.2.3 Reinforcement learning . . . 10
1.1.3 Deep learning . . . 10
1.1.4 Computer vision . . . 11
1.2 Computer vision approaches . . . 13
1.2.1 Artificial neural network . . . 13
1.2.1.1 Artificial neuron . . . 13
1.2.1.2 Activation function . . . 14
1.2.1.3 Feed-forward neural network . . . 14
1.2.2 Convolutional neural network . . . 15
1.2.2.1 Convolution layer . . . 17
1.2.2.2 Activation layer . . . 17
1.2.2.3 Pooling layer . . . 17
1.2.2.4 Fully-connected layer . . . 17
1.2.3 Transfer Learning . . . 17
1.3 Computer vision tasks . . . 18
1.3.1 Camera calibration . . . 19

1.3.2 3-D scene reconstruction 20
1.3.3 Motion analysis 20
1.3.4 Image classification 21
1.3.5 Object detection 23
1.3.5.1 Region-based convolutional network (R-CNN) 23
1.3.5.2 Fast region-based convolutional network (Fast R-CNN) 24
1.3.5.3 Faster region-based convolutional network (Faster R-CNN) 24
1.3.5.4 You only look once (YOLO) 25
1.3.6 Face recognition 26
1.4 Software frameworks 27
1.4.1 OpenCV 28
1.4.2 Darknet 28
1.4.3 TensorFlow 28
1.4.4 PyTorch 29
1.4.5 Caffe2 29
1.4.6 Theano 29
1.5 Evaluation metrics 30
1.5.1 Intersection over union (IoU) 30
1.5.2 Multiple object tracking accuracy (MOTA) 30
1.5.3 Multiple object tracking precision (MOTP) 31
1.5.4 Mean absolute error (MAE) 31
1.5.5 Frames per second (FPS) 31

2 Related work **33**
2.1 Multiple object tracking 33
2.2 Soft-biometrics 34
2.2.1 Body traits 35
2.2.1.1 Height estimation 35
2.2.1.2 Color clothes estimation 36
2.2.2 Facial traits 36
2.2.2.1 Multi-task approaches 36
2.2.2.2 Age and gender estimation 37
2.2.2.3 Facial expression recognition 38
2.2.2.4 Head pose estimation 38

3 Methodology **41**
3.1 Image acquisition 41
3.2 Image pre-processing 43
3.3 Detection and tracking components 43
3.3.1 People detection 44
3.3.2 Track handling 44
3.3.3 State estimation 45

3.3.4 Appearance features 47
3.3.5 Data association 47
3.3.5.1 State metric 47
3.3.5.2 Appearance metric 48
3.3.5.3 Matching 49
3.4 Soft-biometrics extraction 49
3.5 Facial features 50
3.5.1 Age and gender estimation 51
3.5.2 Facial expression recognition 51
3.6 Spatial information 52
3.6.1 Trajectory 52
3.6.2 Height estimation 53

4 Implementation 57
4.1 Module tracking_app 57
4.2 Module detection 57
4.3 Module feature_extraction 57
4.4 Module kalman_filter 58
4.5 Module matching 58
4.6 Module tracking 58
4.7 Module track 58
4.8 Module output_statistics 58
4.9 Module visualization 59

5 Evaluation 61
5.1 Dataset 61
5.2 Hardware and software setup 62
5.3 Tracking results 62
5.4 Soft-biometrics results 63
5.5 Further work 64
5.5.1 Tracking improvements 65
5.5.2 Soft-biometrics extraction improvements 65

Conclusion 67

Bibliography 69

A Acronyms 81

B Media contents 85

List of Figures

0.1 Example of false positive and false negative person detection. Source: [1]. 4
0.2 Cases illustrating tracker-to-target assignments errors. (a) An ID switch occurs when the mapping switches from the previously assigned red track to the blue one. (b) A track fragmentation is counted in frame 3 because the target is tracked in frames 1 and 2, then interrupts, and then reacquires its 'tracked' status at a later point. A new (blue) track hypothesis also causes an ID switch at this point. Source: [2] 5

1.1 Venn diagram of artificial intelligence (AI) and its sub-fields. Source: [3]. 8
1.2 Categories of machine learning tasks. a) Supervised learning. b) Unsupervised learning. c) Reinforcement learning. Source: [4, 5] . 9
1.3 Venn diagram of related fields of computer vision (CV). Source: [6] 11
1.4 Pixel data diagram of simplified grayscale image demonstrating the semantic gap. a) The image itself. b) The pixels labeled with values from 0 to 255, representing their brightness. c) Pixel values by themselves. Source: [7] 12
1.5 Pixel data diagram of a simplified grayscale image. a) Schematic of a single neuron. b) Schematic of a feed-forward neural network. Source: [8] 15
1.6 Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. Source: [9] 16
1.7 CV tasks that are the most popular nowadays. Source: [10]. 18
1.8 Image classification example. Source: [10]. 22
1.9 Image classification challenges. Source: [10]. 22

1.10 Region-based convolutional network (R-CNN) architecture. Source: [11]. 24
1.11 Fast region-based convolutional network (Fast R-CNN) architecture. Source: [11]. 25
1.12 Faster region-based convolutional network (Faster R-CNN) architecture. Source: [11]. 26
1.13 Example of you only look once (YOLO) approach. Source: [12]. . . 27
1.14 Common pipeline for state-of-the-art face recognition systems. Source: [13]. 28
1.15 An example of computing Intersection over Unions for various bounding boxes. The predicted bounding boxes are drawn in red color, while the ground-truth ones are drawn in green color. Source: [14]. 30

3.1 Selected camera equipment for our task. a) Camera Basler ace acA1920-50gc. b) Lens Basler 8mm C125-0818-5M F1.8. Source: [15, 16] 42
3.2 Example of the acquired image and the scene layout. We can see that despite being a quality lens, it contains distortion and vignetting. 43
3.3 Six bounding boxes are detected around the face, but by applying non-maximum suppression, it is possible to remove the redundant ones. Source: [17]. 44
3.4 Example plot of the distance metric values used for track-to-detection spatial displacement cost. 48
3.5 Simplified diagram of single framework pass through. Yellow-marked boxes are optional components used for evaluation. 49
3.6 Example of the 2-D projective transformation. a) Source image. b) Top-down view of the corridor floor generated from (a) using the four corners of a floor tile to compute the homography. c) A planar surface viewed by two camera positions is related by homography H, thus any point can be transformed between these different planes. Source: [18] 54
3.7 Measuring heights in single images. (a) The aim is to compute the height of the human figure relative to the height of the column (reference). The vanishing line of the ground plane has been computed and is shown blue. (b) The unknown height Z_x can be computed from image quantities only as shown in 3.17 (c) A photograph of a garden shed in Oxford. (d) Once the height of the window top edge from the floor has been measured (reference), the height of the man is computed to be 178.8 cm which is about 1 cm off the ground truth. Source: [19] 55

5.1 Image examples of our framework. (a) An example of a person facing the camera so his biometrics can be accurately estimated. Only the height value is inaccurate by 2 cm, which is a very precise outcome. (b) An example of people who are close to each other without affecting the tracker accuracy. (c) An example of people who are even closer than in (b). The person in the middle is fully occluded, and no detection is available for him. However, the red circle is trying to predict his position, in case he reappears in the scene. The prediction of soft-biometrics for the person with ID 13 is inaccurate by one year and 3 cm. 63

List of Tables

5.1 Tracker evaluation table. 62
5.2 Age estimation evaluation table. 64
5.3 Height estimation evaluation table. 64

Introduction

Detection of people and their subsequent identity preservation in a video sequence (tracking) is a part of a more broader domain task called multiple object tracking (MOT). In the basic definition, MOT tries to estimate the position of the objects from multiple predefined classes, and then maintain their identity through the whole video sequence.

In most cases, position estimation is done by a component known as the object detector, which predicts the bounding boxes with real-valued confidences of each object class in each video frame. However, the conventional object detector does not guarantee the relationship between objects in consecutive frames. Therefore, it is necessary to extract additional features from each detected object to build the relationship in the sequence of frames.

The extracted features are mainly based on a visual appearance, movement, and interactions of the object, but complementary information such as camera calibration and known scene parameters can also be incorporated. The subsequent tracking is then achieved by matching detected objects to preserved tracks based on various distance metrics that are calculated between features of the detected objects in a current frame and features of traced objects from previous frames. This approach is also known as online tracking-by-detection which means that only current and previous frame information is available to the tracker, in contrast with offline-based tracking where information from the whole sequence can be used at any time.

One of the tracking benefits is that we can recover the trajectories of the objects that appeared in the video sequence, based on which we can calculate various spatial statistics that can help us to improve existing processes. For example, statistics about the movement within waiting halls might be helpful for companies to improve their indoor space and arrangement.

Even though this thesis focuses only on the task of people tracking, we may still apply many principles from more general MOT. A follow-up step after successfully building the people tracking framework is the extraction of additional class-specific information about people (soft-biometrics). This estimated data can be utilized in many applications from the retail environment; for example, we can utilize them in a retail store where there is the high demand for learning customer trends in specific days and hours, which could be easily achieved by collecting customer information such as age, gender, mood, etc. Other scenarios could be with a personalized advertisement targeted at passers-by in a shopping center or real-time staff alert when senior enters a store to offer immediate assistance. Moreover, we could use this framework to distinguish between customers and employees so we can analyze their interaction or even go further and optimize the distribution of employees around the store.

Motivation

MOT is one of the essential topics in the computer vision field. A large number of surveillance cameras in use has led to strong demand for automatic methods of processing their outputs. For example in the field of crowd image analysis, the scientific challenge is to devise and implement methods for obtaining detailed information about the number, density, movements, and actions involving people observed by a single camera or by a network of cameras.

Historically, the progress in MOT field has been limited by the number and size of the available datasets, and it was especially challenging to make comparisons between algorithms if they have been tested on different datasets under widely varying conditions. Thus, the needs of the researchers eventually formed the very first and most known PETS2009 [20] person tracking dataset. However, this dataset is minimalist. There are only three sequences related to the person tracking with ground-truth information, and evaluation metrics were often applied inconsistently, for example involving using different subsets of the available data, different ways of training the models, or differing evaluation scripts [2].

The big break came in the year 2015 when MOTChallenge [2] was released with the goal of standardization of quantitative benchmark to address such issues. Not only did they create unified evaluation framework with standardized metrics, but they also created the large-scale dataset with a total of 22 sequences, half for training and a half for testing purpose, with a total of 11286 frames or 996 seconds of video. This benchmark has massively transformed the MOT field which resulted in severe improvements to existing algorithms. Its popularity can also be expressed in numbers; for example, 99 MOT tracking algorithms were submitted to the MOT17 challenge [21] during the year 2017 and similarly previous years.

Since the accuracy of existing algorithms is increasing and the price of graphics processing unit (GPU) computations is decreasing, new and more challenging datasets are being invented. VisDrone2018 [22] is a current state-of-the-art MOT dataset with over 260 video clips and more than 2.5 million bounding boxes of various class objects annotated. The frames are captured by several drone-mounted cameras which causes an unusual perspective and makes the dataset even more challenging.

To this date, there are more than ten large-scale MOT benchmarks publicly available which demonstrates that this task, especially with objects such as people, vehicles, and bicycles, has enormous attention in the research community and because the datasets are still updated to be more challenging, there are still places to improve.

The logical extension beyond detection and tracking horizons is the extraction of additional class-specific features. If we would track cars, we could take advantage of estimation of car paint, brand, and type. This information can then be used for various temporal and regional statistics; for example, for estimating the richness of a town by counting luxury-type cars.

If we take another case, which is the extraction of class-specific features about people, we could estimate attributes such as race, gender, height, mood, hair color, and clothes color. These traits are called soft-biometrics, and they are frequently used in cases where we need to complement primary biometric identifiers, such as fingerprint, palm veins, iris pattern, to provide authentication based on the unique identification of the person. The estimation of any class-specific feature is noisy. Therefore, it is essential to have reliable MOT framework, so the features are not collected from a single frame, but appropriately calculated from the whole tracking session.

Although soft-biometric characteristics lack the distinctiveness and permanence to recognize an individual uniquely and reliably and can be easily faked, they provide some evidence about the people identity that could be beneficial [23]. With the use of soft-biometrics, we can differentiate individuals in surveillance video where it is ubiquitous that people are often occluded. In other words, despite the fact they are unable to individualize a subject, they are useful in distinguishing between people, thus maintaining people's identity in a surveillance scene. Another useful utilization is in the retail environment where we can build aggregated statistics such as the number of women between age 25 and 40 visited our store in the morning. If we have such information that in the morning there is 75 % women of visitors, we could utilize this and adapt the store to be more suitable for women, and therefore we will have a better chance to increase profit.

Last but not least, having an accurate and flawless MOT framework is crucial for further expansion to popular multi-target multi-camera tracking (MTMCT) field, which is the problem of determining who is where at all times given a set of video streams as input. The output of this task is also a set of person trajectories but from a wider area. Person re-identification (ReID) is a closely related problem in this field. Given a query image of a person, the goal is to retrieve from a database of images taken by different cameras the images where the same person appears. [24]

Challenges

MOT field is deeply explored – many methods have been proposed, and many surveys have been conducted [25, 26, 27]. However, it is still considered as not successfully solved computer vision task. In other words, a saturation point has not yet been reached.

MOT task is an extension of object detection from single images to video sequence. The main challenges when using an object detector for tracking are that the resulting output is unreliable and sparse, i.e., detectors only deliver a discrete set of responses and usually yield false positives and false negatives (missing detections) as shown in Fig 0.1. So, in addition to such object detection errors, identity switches are frequent in any MOT framework (see Fig. 0.2).

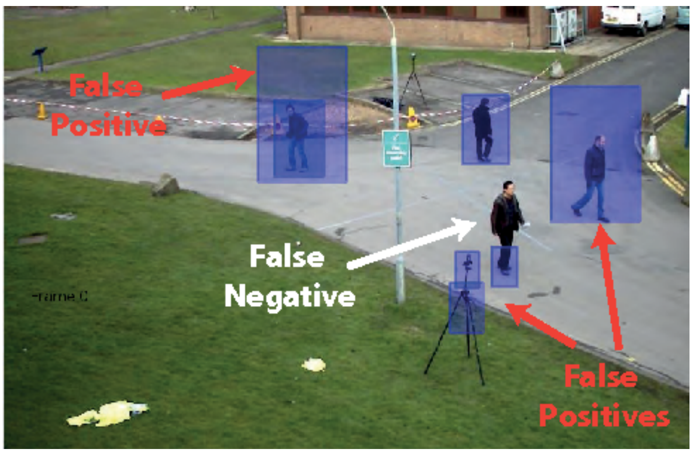


Figure 0.1: Example of false positive and false negative person detection. Source: [1].

ID switches occur when there are two object trajectories are produced for one ground-truth object. Fragmentation occurs when there are two trajectories are produced with a time gap between them for one ground truth object, which implies that detections are missed in several frames. In theory, the robust tracker should handle both of these flaws. It should fill gaps in detections by propagating information from neighboring frames, and it should also filter false positive detections based on the information from previous frames.

However, it has been found out during multiple object tracking challenges that in practice, it is not entirely so. For example, in MOT [21] dataset from 2016, 18 % of tracks are not covered by detections at all, and 37 % percent of tracks are covered only by low confidence of detections. When there is no high-quality detection for particular ground-truth track, then the tracker cannot resolve this problem at all, which implies that tracker usually reduces only false positives and raise false negatives by removing low-confidence detections.

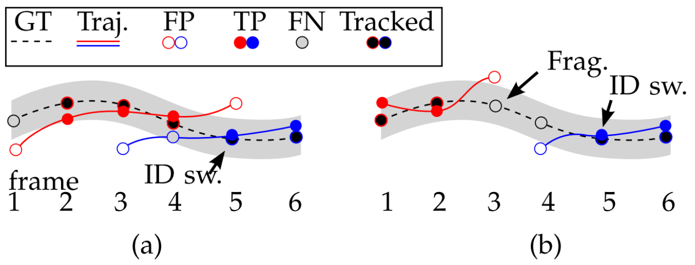


Figure 0.2: Cases illustrating tracker-to-target assignments errors. (a) An ID switch occurs when the mapping switches from the previously assigned red track to the blue one. (b) A track fragmentation is counted in frame 3 because the target is tracked in frames 1 and 2, then interrupts, and then reacquires its 'tracked' status at a later point. A new (blue) track hypothesis also causes an ID switch at this point. Source: [2]

Nowadays, many researchers state that a robust object detector is a key to good tracking [21, 28, 29] and during recent work [25, 26, 30], it has been shown that modern object detectors can locate object even in complex, crowded scenes. However, false positives have remained frequent.

If we narrow down to the tracking people, it is even more difficult because they are very dynamic objects. People tend to change position, direction, and posture often, but they also have different height and body proportions. In real-world scenes, long-term occlusions are also frequent. As a result, it is vital for people tracking systems to be flexible so that it can handle as many different situations as possible.

Lastly, it is a topic from computer vision field where most of the work is done over images represented by matrices. It is well known that working with images is computationally demanding because each change needs to be applied for each pixel and although the image algorithms can be well parallelized, we still need to keep in mind the computation costs.

Objectives

The goal of this thesis is to design and implement a sophisticated framework that could utilize surveillance sequences for people online tracking followed by the extraction of as much data as possible about the people in the scene, which is a three-step process. First, we need effectively and precisely acquire people detections in each frame. Then, we need to utilize a robust tracking algorithm that will maintain people identities. Lastly, gathered data from people tracking sessions must be appropriately processed for useful output statistics.

Assumptions

The detection and tracking of people in surveillance footage is a broad topic. For this reason, the work is limited only to the one static camera watching a known scene. We assume that the captured scene is entirely under our control which means we can make any measurements and calibrate the camera. Moreover, we take into account the requirements for online processing so that the framework can be improved for real-time inference in the future.

Thesis structure

The rest of this thesis is organized as follows. In the first chapter, we present a theoretical background which is crucial for the understanding of solving this task. Chapter 2 is devoted to the related work. Algorithm design and proposals are presented in chapter 3. Implementation details are explained in chapter 4, followed by the evaluation presented in chapter 5. The whole work is wrapped up in the last Conclusion chapter.

Chapter 1

Theoretical background

This chapter contains the necessary summary of theoretical knowledge to understand this work. We start with a general overview of artificial intelligence areas, emphasizing the computer vision field and its approaches with follow-up tasks. Lastly, we try to address the metrics needed to evaluate the algorithms used in the thesis. A reader who is familiar with the thesis topic can continue straight to the next chapter.

1.1 Relevant areas

In this section, we go through a brief explanation of topics such as artificial intelligence, machine learning, deep learning, and computer vision. The reason for this is that many people consider these terms as effectively synonymous, but each one deals with different tasks.

1.1.1 Artificial intelligence

AI is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans and to this date, researchers cannot agree on its precise definition [31]. The main characteristic is that, unlike traditional algorithms, the algorithms of artificial intelligence are capable of learning from new and past data. They can enhance themselves by learning new strategies, or by improving existing algorithms.

Although creating a general artificial intelligence that is comparable to human has proved to be extremely difficult, over the past fifty years researchers have developed a set of procedures that achieve partial success in AI sub-fields such as expert systems, genetic programming, state space search, data mining, machine learning, deep learning, and computer vision [32].

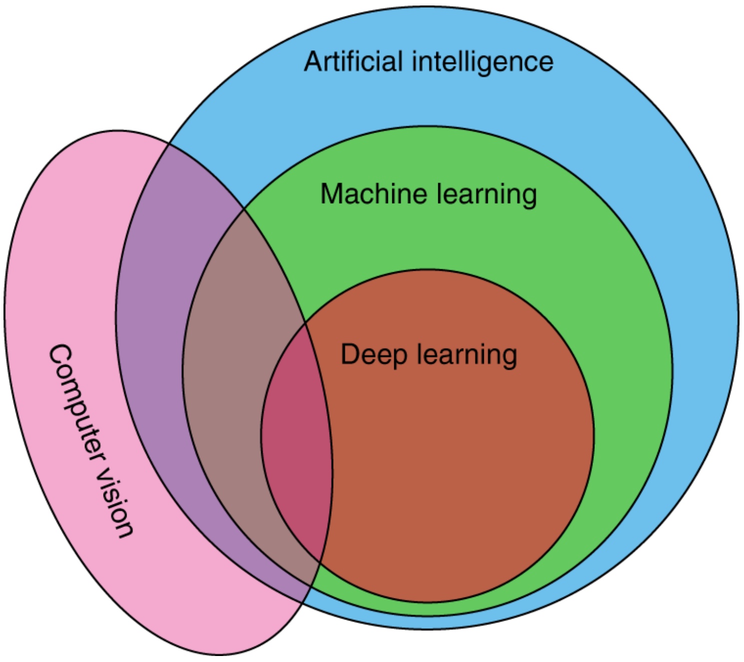


Figure 1.1: Venn diagram of AI and its sub-fields. Source: [3].

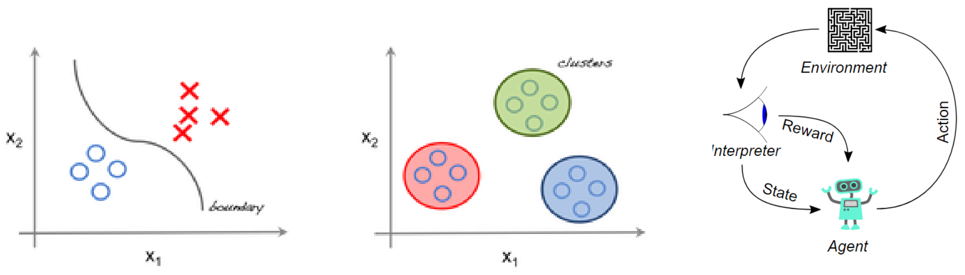
1.1.2 Machine learning

While the AI is a broader concept, machine learning (ML) is the core part of it. The popular definition from [33] states: *"Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions."*

ML is sometimes considered difficult to understand. However, the underlying intuition is not that complicated. At its core, we can imagine that the ML algorithm is just searching a decision hyperplane of best fit in many dimensions. If we have up to 2-D feature space of the problem, we can even visualize the split and the results. That is why dimensionality reduction such as principal component analysis (PCA) [34] is often used.

The complexity in the machine learning tasks is the training and preparation of the training dataset. There is a great emphasis on the flawlessness of the dataset because if it contains errors, the algorithm can quickly learn these imperfections and thus decrease the generalization performance [35].

There are three [36] main ML categories that are being actively researched – supervised learning, unsupervised learning, and reinforcement learning. They differ from each other in the way they handle data, but also with the outputs they provide.



(a) (b) (c)

Figure 1.2: Categories of machine learning tasks. a) Supervised learning. b) Unsupervised learning. c) Reinforcement learning. Source: [4, 5]

.

1.1.2.1 Supervised learning

In supervised learning [35, 36, 37], the algorithm focuses on building a mathematical model from a set of available data that contains both the inputs and also the desired outputs. The most common example is to predict house prices based on various features such as the number of rooms, square floor feet, lot area, general condition, etc. ML model takes these features as inputs and based on its internal parameters, it produces output price prediction. This type of supervised learning is called regression. It is characterized by that the output is continuous value.

Another type of supervised learning is classification which is used when the outputs are restricted to a limited set of values. We could consider a simple classification problem as identification of spam emails based on its content. The output prediction for this task would be of either ”spam” or ”not spam.” A traditional supervised classification model used for decades is a support vector machine (SVM) [38] that divides the data into regions separated by a linear boundary.

There is also a popular sub-category called semi-supervised learning [39] that emphasizes on using incomplete data during training, in terms of missing labels for some data samples.

1.1.2.2 Unsupervised learning

Unsupervised learning algorithms [4, 35, 36] try to build a mathematical model to find patterns in data. The data given to the unsupervised model in training phase contains only input features and no desired output labels. Algorithms are then left to themselves to explore and discover critical structures in the data and group the inputs into clusters (categories). In the previous example with houses, the model would be able to find similar houses for input parameters, but not to predict the price of a given house.

Typical algorithm representatives in unsupervised learning are hierarchical clustering [40], k-means [41], and DBSCAN [42].

1.1.2.3 Reinforcement learning

"Reinforcement learning is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment. [43] The learning is achieved based on the positive or negative reward for agent's actions in the environment. Given the agent's and environment's states, the agent only takes actions which will maximize his reward or will explore a new possibility. Results of these actions are then fed back into the agent, and based on that it will change its state. This step is repeated many times to improve the agent's behavior for future decisions.

Examples of rewards can be winning a game, scoring more points or earning more money. Thus it fits well dynamic tasks. Reinforcement learning, especially with deep neural network extension [44], already performs well on a small human-involved task such as playing Atari games and it presents current state-of-the-art results in this field.

1.1.3 Deep learning

Deep learning (DL) [45] is a specialized form of ML, and it is currently the cutting edge of what machines can do. Its methods have far surpassed any previous traditional algorithms for classification of images, text, and voice and we can find its use in all current machine learning applications.

The essential advantage is that unlike ML models, DL models are trained by large labeled sets of data where the model architecture learns the domain features directly without the need for manual feature extraction. This is also known as "end-to-end learning" where a model is given data, optimization criteria, a task to perform and it learns how to solve this task automatically. Another important fact is that today's DL models often continue to improve as the size of the data increases [46].

DL tasks can be categorized into the same categories as ML – supervised, unsupervised, and reinforcement learning. The most common DL models are based on an artificial neural network, which is why deep learning models are often referred to as deep neural networks, but other architectures such as convolutional neural networks (1.2.2), deep belief networks [47], and recurrent neural networks [48] are also popular. Unlike traditional neural networks with a few hidden layers, deep networks can have hundreds of layers with thousands of neurons in each of them.

Although many DL concepts such as neural networks, backpropagation, gradient descent had been around for decades [49, 50, 51], some researches asses that victory of Krizhevsky in October 2012 ImageNet competition started the "deep learning revolution" [52]. However, it was not the only big break that happened, for example, Google's Deep reinforcement learning based AlphaGo beating the best Go player in the world [53] also had a great impact on the research community.

There are two main drawbacks while using DL architectures. Firstly, it requires a large amount of labeled data, which is an even bigger problem in critical applications. In an example such as autonomous driving, it is often required to have thousands of hours of video, which can take up a few petabytes of storage space. Secondly, DL requires substantial computing power, which is most often achieved by using high-performance GPUs that have parallel architecture. However, most people, especially researchers, do not have their GPU cluster, so they use cloud computing that enables them to reduce training time from weeks to hours.

There are also many criticism [46, 54] concerns around DL because methods are often looked at as a black box with the lack of theory, where most confirmations are done empirically rather than theoretically.

1.1.4 Computer vision

CV is the process of using computers to process, understand and analyze various types of image data in order to produce numerical or symbolic information, e.g., in the forms of decisions [55]. The motivation is to automate tasks that human visual systems can do to utilize human resources on more critical tasks. Moreover, with a large number of surveillance cameras, it is not possible to monitor all of them by people.

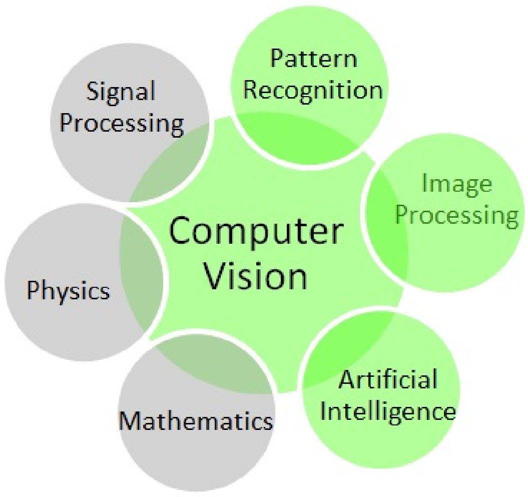


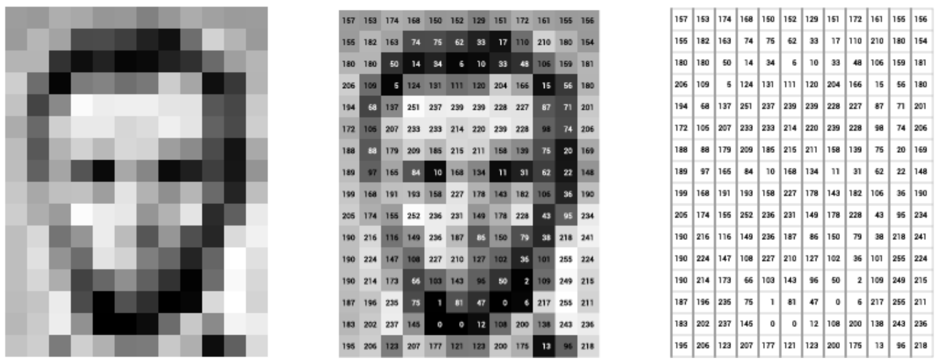
Figure 1.3: Venn diagram of related fields of CV. Source: [6]

.

The whole process from analyzing image data to subsequent decision making is called a CV pipeline. The traditional pipeline starts with image acquisition, followed by image preprocessing and feature extraction step, ending with a classifier. This pipeline follows well-established principles from any ML task. Hence we do not consider it necessary to explain it in this work, but we provide a reference to [56] which offers a detailed description of each step.

The analyzed image data can be a standard photo or video sequence from a common red-green-blue (RGB) camera, but also to more complex data such as multi-camera video sequence and multi-dimensional imagery from satellite or medical equipment.

Computers interpret this visual content very straightforwardly – as a series of pixels that forms a matrix, where each pixel has its own set of color values (see Fig. 1.4). However, in order for the machines to make decisions, they need to understand higher-level concepts in the data, not to use just plain pixel values. For this reason, and also because the size of the imagery data is large, feature extraction is a common technique to obtain high-level features, thus helping computers to understand the visual context [7].



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

(a) (b) (c)

Figure 1.4: Pixel data diagram of simplified grayscale image demonstrating the semantic gap. a) The image itself. b) The pixels labeled with values from 0 to 255, representing their brightness. c) Pixel values by themselves. Source: [7]

.

To this date, we can observe two main approaches in CV pipelines – hand-crafted ML methods and DL methods [45]. The main difference is how features are obtained before they are fed into a classifier. We can think of mean, variance, median, min, and max of our data as useful measures for discrimination of samples, but since these features are specified explicitly, they are called hand-crafted. In practice, we have more complicated tasks that require more sophisticated statistical functions and feature extractors such as Haar [57], local binary pattern (LBP) [58], histogram of oriented gradients (HOG) [59], Scale-invariant feature transform (SIFT) [60], Speeded-Up Robust Features (SURF) [61], but the idea is the same – we know how the features will look like in advance. In DL, we know nothing about the features until we learn them from data itself. We still need to specify model and its parameters, but the real feature learning is achieved by an iterative optimization process.

CV is nowadays ubiquitous in our society. It is used in applications such as image understanding, medicine, self-driving cars, augmented reality, and drones and there are many sub-domains of CV that are actively researched [55]. The most significant interest is currently on scene reconstruction, object detection and tracking, human pose estimation, and style transfer. It may be a surprise, but the core of many applications often builds upon something simple as image classification, and while many types of CV algorithms have been around since the 1960s, recent developments in computing capabilities have driven significant improvements in how well machines understand the image content. Furthermore, with the advent of DL approaches, CV is becoming increasingly popular, and at the same time, there is a noticeable increase in the accuracy of many existing CV applications [45].

1.2 Computer vision approaches

Because the CV field has massively transformed into DL algorithms, we present a brief overview of some conventional approaches that are used for solving today's CV tasks.

1.2.1 Artificial neural network

Artificial neural network (ANN) [45, 62], or simply neural network (NN) in this context, is an interconnected graph of artificial neurons that uses a computational model for solving ML tasks . In more practical terms, NN is non-linear statistical data modeling or decision-making tool that can be used to model complex relationships between inputs and outputs or to find patterns in data. In most cases, a NN adapts its internal parameters based on information that flows through the network. Thanks to many breakthrough results in recent years, it generated much excitement in the research community [52].

1.2.1.1 Artificial neuron

There are many types of NNs [45]. However, the most researched and utilized is a particular type known as the multi layer perceptron (MLP) [63]. Its basic unit of computation is called neuron (or node). It receives inputs from other neurons and computes an output value. Each input x_i of a neuron has its a particular weight w_i value associated. The weight can be understood as relative importance to other neuron inputs. Weights and inputs are added together in weighted sum, and additional bias parameter b is used as a correction. In the end, activation function φ is used to generate an output of the neuron y_k. In mathematical notation, the output of a neuron is given by:

$$y_k = \varphi\left(\sum_{j=1}^{n} w_j x_j + b\right) \tag{1.1}$$

1.2.1.2 Activation function

It is very common that applications from the real world are non-linear, and in order to find an approximate solution to them, it is necessary to include non-linearity into the prediction model [45]. According to the Universal approximation theorem [64], by incorporating non-linear activation function into an MLP architecture with single hidden layer and a finite number of neurons, it is possible for the model to approximate any continuous function, i.e., can in principle learn anything.

To this date, many activation functions were proposed, but the most commonly known [10] are:

- **Sigmoid** – The sigmoid non-linearity squashes real numbers to range between $[0, 1]$.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.2}$$

- **tanh** – The tanh non-linearity squashes real numbers to range between $[-1, 1]$.

$$\tanh(x) = 2\sigma(2x) - 1 \tag{1.3}$$

- **ReLU** – The activation value of ReLU is simply thresholded at zero. As the ReLU has proved to work very well, several other variants such as Leaky ReLU, ELU, and SELU have emerged.

$$f(x) = \max(0, x) \tag{1.4}$$

The activation function selection is crucial for task accuracy and performance. However, their thorough description is very comprehensive. Thus we recommend [10] for their detailed overview.

1.2.1.3 Feed-forward neural network

Feed-forward NN [36, 45] contains multiple neurons arranged in layers. Nodes from adjacent layers have connections between them, and all these connections have associated weights. In this type of network, the information during prediction moves only in the forward direction, hence the name. No cycles are allowed in this architecture.

A feed-forward NN can consist of three types of layers:

- **Input layer** – The input layer is always at the beginning of the network. It provides information from outside of the model to the network. No computation is performed at this stage, and values are just distributed to further neurons.
- **Hidden layer** – Neurons in this layer perform computations to output nodes based on input nodes and internal parameters. A feed-forward network can have zero or multiple hidden layers.
- **Output layer** – This layer is responsible for transferring the computed information from the network to the outside world. The transferred information usually represents a class probability score or some real-valued prediction.

The weights in neurons are iteratively improved in the training phase, where data is forwarded through the network, then the difference between ground-truth and the predicted value is calculated. The procedure to improve the weights is known as backpropagation [65]. An example of a feed-forward neural network is shown in Fig. 1.5. Feed-forward NN with many hidden layers is called deep NN.

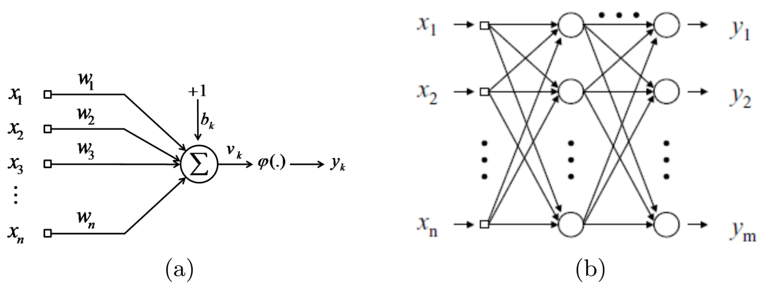


Figure 1.5: Pixel data diagram of a simplified grayscale image. a) Schematic of a single neuron. b) Schematic of a feed-forward neural network. Source: [8]

.

1.2.2 Convolutional neural network

One of the most common methods for solving computer vision task is convolutional neural network (CNN) [66, 50, 45]. It uses convolutional layers to learn features from input data without minimal preprocessing. Therefore, it can eliminate the need for manual feature extraction as in traditional methods. The features are learned while the CNN trains on large labeled image dataset.

This approach has shown to be highly accurate since it outperforms humans in image recognition task [52] and it is one of the most popular techniques for deep learning. Current applications such as object self-driving cars, object detection, medical image classification, combined with advances in GPUs and parallel computing, heavily relies on CNN architectures.

A CNN can have tens to hundreds of layers that each learn to detect different patterns or features of an image. Depth of the CNN is a critical component for good performance [52]. Since there is the explicit assumption that the inputs are grid-like topology, such as an image, certain properties are encoded into layer architecture that makes the forward function more efficient by reducing the number of parameters in the network. Network layers perform operations that alter the data with the intent of learning features specific to the data.

Three of the most critical layers are convolution (1.2.2.1), activation (1.2.2.2), and pooling (1.2.2.3), which are often applied multiple times in a row before concluding the process of feature extraction. The goal of repetition is to identify different features, and the argument for this is the observation that images contain hierarchical structure (e.g., faces are made up of eyes, which are made up of edges, etc.), so several layers of processing will increase in extracted features complexity to features that uniquely define particular object. The outputs of this whole process are then passed into a fully connected layer for final output, i.e., class score probabilities.

Layout, number, and type of layers form the architecture of the CNN. To this date, dozens of architectures were proposed, and the most famous are LeNet [51], AlexNet [66], VGGNet [67], Inception [68], ResNet [69], ResNeXt [70], and DenseNet [71]. Since each of these architectures would require extensive description, we consider it as out of the scope of this thesis, and we suggest [10, 72, 73] for further reading.

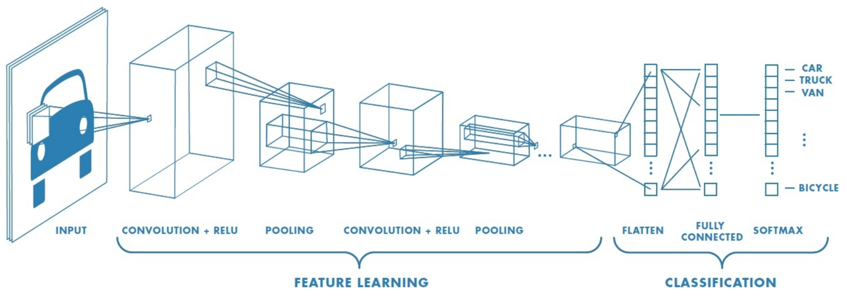


Figure 1.6: Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. Source: [9]

.

1.2.2.1 Convolution layer

The convolution layer is the core building block of any CNN, where the input image pixels are modified by a convolutional filter, which is a matrix that is multiplied with different parts of the input image. The filter is spatially smaller than an image (e.g., 3x3, 5x5, 8x8), but is more in-depth. Each filter aims to activate certain features from the images. The output of this layer is known as a feature map (alternatively activation map), and it is usually smaller in size but has more dimensions. Theoretically, a feature map will be less redundant and more informative than the original input.

1.2.2.2 Activation layer

The purpose of this layer is the same as in the ANN, to introduce non-linearity in the feature maps. Favorite choice in the CNN case is ReLU due to its simplicity which implies an effective training process.

1.2.2.3 Pooling layer

Since the convolutional layer expands dimensionality, pooling is a process that reduces the size of the feature map by a factor of whatever size is pooled. The consequence is the reduction of the number of parameters that the network needs to learn. The input image is scanned over each dimension by a sliding window and either the max, sum or average the window is taken as a representation of that portion of the image.

1.2.2.4 Fully-connected layer

A fully-connected layer is most often the last layer of the CNN architecture. It is implemented as a common feed-forward NN with fixed input size. Adding this type of layer usually helps with combining the high-level features from convolutional layers into a non-linear function. A frequent choice of fully-connected layer output type is a vector that contains the probabilities of each object class of an image being classified.

1.2.3 Transfer Learning

It is sporadic for people to train an entire CNN from scratch. There are three main reasons for this [10]. Firstly, it has been proved that CNN and NN are very sensitive to proper weights initialization. Many work [74, 75] have been done on this topic, and it is generally not recommended to use random or zero initialization since it can lead to vanishing or exploding gradients during training. Secondly, there are very few datasets with sufficient size. Using only a tiny dataset will lead to insufficient accuracy or overfitting. Lastly, the training phase is computationally intensive. Modern CNN architecture can take a few weeks of training on multiple GPUs to converge.

In practice, it is common to take pre-trained parameters from existing CNN used for a similar task (e.g., ImageNet classification [52], which contains 1.2 million images with 1000 categories) and use them as initialization or a fixed feature extractor for the task of interest. This use of existing CNN as a feature extractor is straight-forward because only the last fully-connected layer needs to be removed or replaced [10].

1.3 Computer vision tasks

In this section, we provide a quick overview of tasks relevant to the thesis topic and their specific challenges. Since the first early steps of computer vision field in the 1960s, the scientists tried to build camera robots with intelligent behavior [32, 36, 55]. However, it turned out to be a much more complex problem than they initially thought and as a first step, they desired to extract 3-D structure from 2-D images to achieve full scene understanding. Therefore, forming the very first computer vision task – scene reconstruction (1.3.2). Moreover, it was this time, that many feature extraction algorithms that are popular today were founded, including edge detector and lines extractors [55].

Later, the robots, which partially understood the scene, needed to move around while avoiding obstacles, which could be accomplished through motion analysis (1.3.3). Some existing algorithms such as Kalman filter [76] already existed. Hence they were adapted to CV field. However, motion detection algorithms such as optical flow [77] and background subtraction methods [78] have also been developed.

In recent years, there is a significant advance in this field, and we can finally see the real applications of these robots which are, for example, self-driving cars [79]. As a result, today's most considerable emphasis is on feature-based methods used together with machine learning algorithms to produce numerical or symbolic information for decision making [45]. It should be noted that many algorithms still come from the research done in early beginnings.

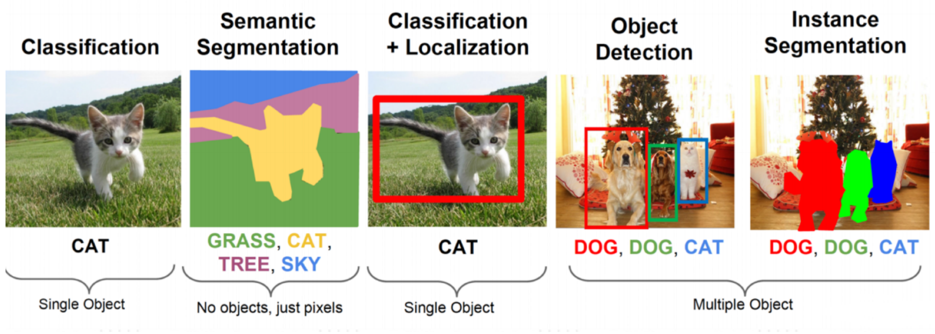


Figure 1.7: CV tasks that are the most popular nowadays. Source: [10].

1.3.1 Camera calibration

Cameras and other sensors with optics have been on the market for decades. Recently they became the part of our everyday life which can be demonstrated, for example, by finding it on every phone or laptop. However, these cameras have one big drawback caused by their easy availability and cheap purchase price. Their image is distorted considerably. Fortunately, the distortion can be described by mathematical equations. Therefore, it can be partially or entirely removed.

Since distortion degrades the image quality, it is recommended to remove it before applying any algorithms [55]. The most popular technique to remove distortion is camera calibration [80], which is the process of finding the intrinsic and extrinsic parameters of arbitrary camera setup. The intrinsic parameters deal with the internal characteristics of the camera and lens combination. The extrinsic parameters refer to the 3D orientation and position of the camera in the space. Example of intrinsic parameters is the focal length, principal point, sensor skew. With knowledge of intrinsic parameters, it is possible to establish a relation between image pixels and real-world units to do real-world measurements and also remove lens distortion, which degrades image quality.

Two essential types of distortion exist. First is radial distortion that causes straight lines to appear curved. It becomes larger the farther points are from the center of the frame. The other one is tangential distortion caused by the lens not parallel to the image plane. Before running any critical CV application, both distortions need to be corrected first.

The radial distortion can be represented as follows:

$$x_{distorted} = x(1 + k_1r^2 + k_2r^4 + k_3r^6) \tag{1.5}$$

$$y_{distorted} = y(1 + k_1r^2 + k_2r^4 + k_3r^6), \tag{1.6}$$

and tangential as follows:

$$x_{distorted} = x + [2p_1xy + p_2(r^2 + 2x^2)] \tag{1.7}$$

$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2xy], \tag{1.8}$$

where x, y are coordinates of the original point, $x_{distorted}, y_{distorted}$ are their distorted projections, k_1, k_2, k_3, p_1, p_2 are unknown distortion coefficients, and r is a radial distance of point x, y from the center of the image. The distortion coefficients do not depend on the scene viewed, and they remain the same regardless of the captured image resolution.

To find the distortion coefficients, we must provide several images of predefined and well-known pattern to camera calibration framework. Generally, it is recommended to use at least ten images of the pattern from different angles. The framework then detects specific points in the image of which relative positions are known. Based on these image coordinates and their relative correspondence in the real world, it is possible to solve equations mentioned above and get distortion coefficients. A standard pattern used in camera calibration is a chess board with square corners as target points. For a more detailed description of this technique, we recommend [80].

1.3.2 3-D scene reconstruction

3-D reconstruction is the process of obtaining 3-D information from the captured 2-D scene [18]. It is a complicated process by the fact that in the imaging process the depth of the space is lost. To solve this task, stereo vision or methods based on triangulation with multiple captured images from different angles are often employed. However, some single-view approaches are sufficient to obtain a partial or complete reconstruction, and for the scope of this thesis, they are the most relevant.

Single-view approaches might be further categorized into calibrated and uncalibrated methods [81]. Calibrated ones use parameters obtained by camera calibration presented in 1.3.1, and their main disadvantage is that internal camera parameters may not always be constant. The intrinsic parameters may be affected by environment effects or only by changing a focal length. Proposals for the calibrated methods can be found in [82, 83, 84].

On the other hand, algorithms developed for uncalibrated scenarios require no knowledge of the camera's intrinsic and extrinsic parameters. The use of known scene constraints replaces camera calibration. These constraints include planarity of points, the parallelism of lines, and parallelism of points. Therefore, no scene markers or specialized sensors are required; these cues are inferred directly from the captured 2-D image, which leads to flexible algorithms that can be applied to a wide range of scenarios. One of the most famous works in this field is [85].

Further, the discipline that deals with estimating real-world measurements in 2-D image scenes is known as single-view metrology. The groundbreaking work in this field is [19].

1.3.3 Motion analysis

As a result of access to a massive amount of video data, motion analysis [55] has drawn considerable interest in recent times. Its main goal is to output information based on the apparent motion in the sequential images. The information produced often depends on neighboring images and is related to specific time-point.

Motion analysis [55] is a field that has been researched since the 1970s, but since then, the algorithms, approaches, and also disciplines has changed. Most proposed methods are based on pixel displacements of underlying physical points. The simplest representative in this field is an optical flow algorithm [77] that detects motion. Nowadays we are more likely to encounter more complex frameworks used in the task such as human motion analysis, scene behavioral analysis, and surveillance tracking.

In the context of tracking, motion information is mostly employed as one of the features that are used during the matching step between new and previously tracked objects. Modern motion filters also allow us to predict the movement information of objects even if no motion information is available in multiple frames [86]. Examples of well-used motion filters are particle filter, and Kalman Filter and its non-linear variants. Their thorough comparison can be found in [86].

1.3.4 Image classification

The typical problem in computer vision field is determining if image data contain a specific object. Image classification [52, 66, 45], with its main component known as classifier, is a particular application of computer vision which assigns an input image one label from a fixed set of categories. The general approach is to assign probabilities for each class and then choose the most likely one. Despite its simplicity, it is one of the core problems in computer vision with a large variety of applications. Moreover, many other distinct computer vision tasks (localization, object detection, segmentation, etc.) can be reduced to image classification.

In Fig. 1.8, we can see the image of a cat with associated probabilities of belonging into four categories. We need to keep in mind that in this figure, the image is represented as a large 3-dimensional matrix of numbers in a computer's memory. In this case, the image is 248 pixels wide, 400 pixels tall, and has three color channels. Therefore, the image consists of $248 \times 400 \times 3$ numbers or a total of 297600 numbers. Each number is an integer that ranges from 0 (black) to 255 (white). To conclude, Therefore, the task is to turn a quarter of a million numbers into a single label, such as cat. [10]

Since this task of recognizing objects is relatively trivial for humans, it only makes sense to consider challenges for CV algorithms. The main challenges are visualized in Fig. 1.9 and according to [10] they are especially:

- **Viewpoint variation** – A single instance of an object can be oriented in many ways concerning the camera.
- **Scale variation** – Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image).
- **Deformation** – Many objects of interest are not rigid bodies and can be deformed in extreme ways.
- **Occlusion** – The objects of interest can be occluded. Sometimes only a small portion of an object could be visible.

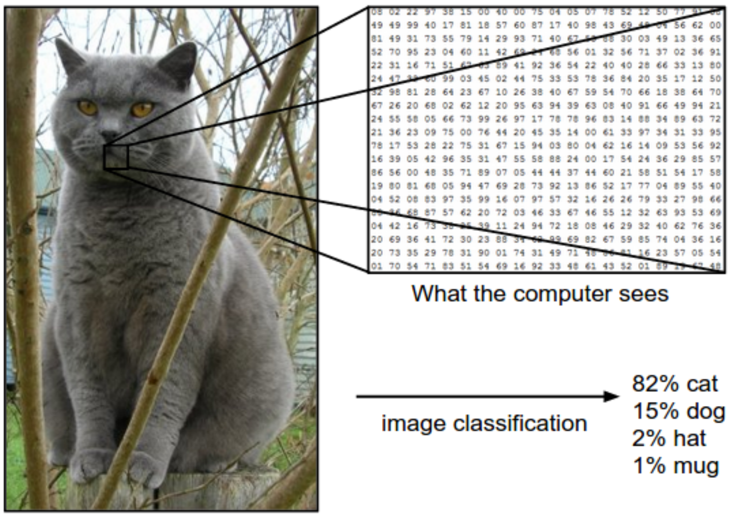


Figure 1.8: Image classification example. Source: [10].

- **Illumination conditions** – The effects of illumination are drastic on the pixel level.
- **Background clutter** – The objects of interest may blend into their environment, making them hard to identify.
- **Intra-class variation** – The classes of interest can often be relatively broad, such as a chair. There are many different types of these objects, each with their appearance.

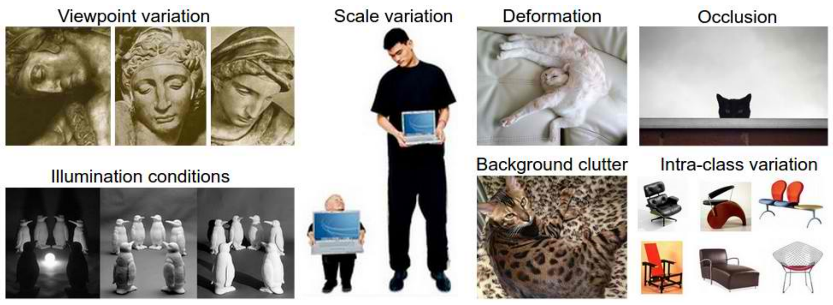


Figure 1.9: Image classification challenges. Source: [10].

A good image classification model must be invariant to the cross product of all these variations, while simultaneously retaining sensitivity to the inter-class variations [10].

1.3.5 Object detection

Image classification models can classify input images into the most likely category. However, typical images including photos are usually more complex and contain multiple objects. Hence, assigning single object category does not make much sense.

Object detection [11, 87] is a well-researched and more appropriate method that helps to identify multiple objects from predefined categories from a single image. The output of the typical object detector is an object's bounding box, category, and confidence of the detection. The bounding box patches can be then used for further visual tasks.

The big break in object detectors came in 2001 with the introduction of Haar cascades by Paul Viola and Michael Jones [57]. Their detector was able to operate in real-time and was subsequently implemented in all digital cameras. Although it could be trained to detect a variety of object classes, it was used mainly for the face detection task.

After a long time, another promising detector known as HOG [59] was introduced in 2005. HOG was focused on pedestrian detection in static images and was further improved for video sequences, as well as to a variety of object classes including animals and vehicles. It has been used for a long time in conjunction with other feature extractors for various detection tasks until the success of DL architecture in 2012 ImageNet competition [52]. Since then, the use of conventional detectors is mostly replaced with deep neural networks.

In object detection based on CNNs, there are two main core design choices [45]. First is, hypothesize object regions and then classify them. Second, divide the image into a grid and directly predict bounding boxes with class probabilities in a single evaluation, thus only one feed-forward pass. Although a few state-of-the-art representatives for each category are briefly reviewed below, we suggest [11, 88, 87] for their detailed overview.

1.3.5.1 Region-based convolutional network (R-CNN)

R-CNN [89] is very first and intuitive architecture that started the era of object detection with CNNs. The pipeline of the model begins with scanning the input image for possible objects using Selective Search algorithm [90], which generates a large number of proposals that are reduced to some reasonable amount (typically to order of thousands). Moreover, each proposal is also resized to match the input of a CNN. Further, CNN is used to extract image features, and the output vector is then fed into an SVM [38] classifier to verify if an object exists within the region. If yes, a linear regressor is used to refine the position of the bounding box.

To sum it up, this approach turns object detection into an image classification discussed before. The drawback of this method is that the training

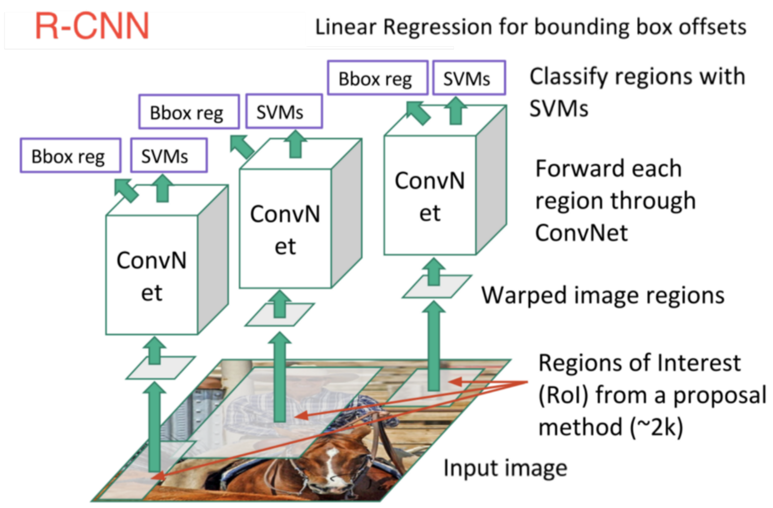

R-CNN
Linear Regression for bounding box offsets
Bbox reg
SVMs
Classify regions with SVMs
Forward each region through ConvNet
ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

Figure 1.10: R-CNN architecture. Source: [11].

and inference performance is very slow. For example, the inference time for a single image varies from tens of seconds to minutes.

1.3.5.2 Fast region-based convolutional network (Fast R-CNN)

Just a year after the publication of R-CNN, the same authors improved the approach with new Fast R-CNN [91] architecture. It resembled the original in many ways. However, they drastically improved the training and testing speed performance.

The improvement consisted of two main aspects. The feature extraction run by CNN is no longer run thousands of time over each of the proposed regions, but only once over the entire image. The regions are still obtained by Selective search algorithm [90]. However, its input is the feature map output of the CNN. The proposed regions are then reshaped using an region of interest (RoI) pooling layer and instead of training many different SVM [38] classifiers for each object class, there is a single fully connected NN with softmax layer that outputs the class probabilities directly.

1.3.5.3 Faster region-based convolutional network (Faster R-CNN)

Both of the architectures mentioned above use a Selective Search algorithm [90] to find out the region proposals. However, it turned out that Selective Search is the computational bottleneck. Besides that, it has another big disadvantage – it is a fixed algorithm; no parameter learning is happening during the training phase which may lead to bad region proposal candidates. Therefore,

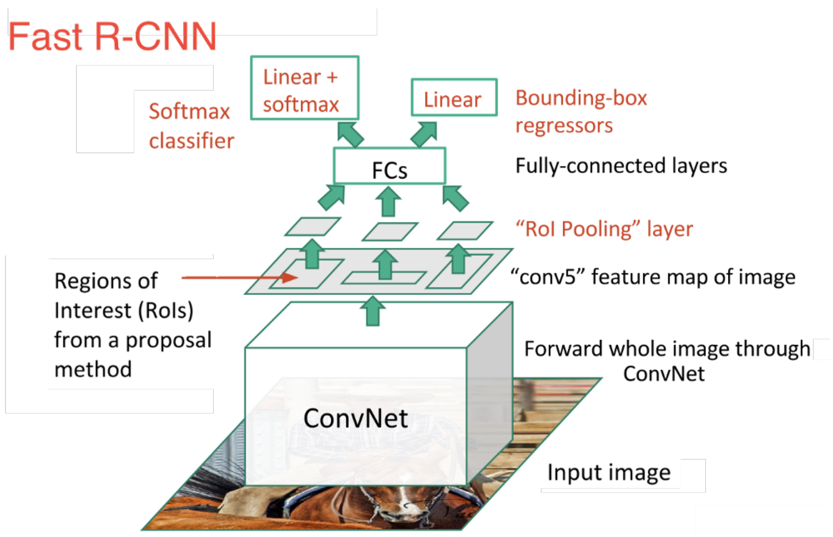


Figure 1.11: Fast R-CNN architecture. Source: [11].

a new architecture known as Faster R-CNN [92] was introduced to improve these shortcomings.

Faster R-CNN completely replaced the use of the Selective Search algorithm with a separate NN (known as the region proposal network (RPN) [92]) to identify region proposals. The RPN is run right after the feature extraction, and once we obtain the region proposals, they are feed into what is essentially Fast R-CNN.

To conclude, Faster R-CNN is a combination of the RPN with Fast R-CNN. It is much faster than its predecessors, and it can even be used for real-time object detection in a video sequence. It is still widely deployed in today's frameworks thanks to its speed and accuracy performance.

1.3.5.4 You only look once (YOLO)

All of the previous object detection algorithms use proposed regions to localize the object within the image. It means that the detector does not look at the whole image. Instead, it evaluates regions which have high probabilities of containing the object. In YOLO [12], a single CNN directly predicts bounding boxes and class probabilities with a single forward pass.

Initially, the model takes the input image and divides it into a $S \times S$ grid. Within each cell of the grid, m bounding boxes are taken. For each of selected bounding box, the network outputs a class probability and refinement for the bounding box. If the bounding box has the class probability above a threshold value, it is then used to locate the object within the image.

Since the model predicts a high number of bounding boxes, the non-maximum suppression (NMS) [17] procedure is applied at the end of the net-

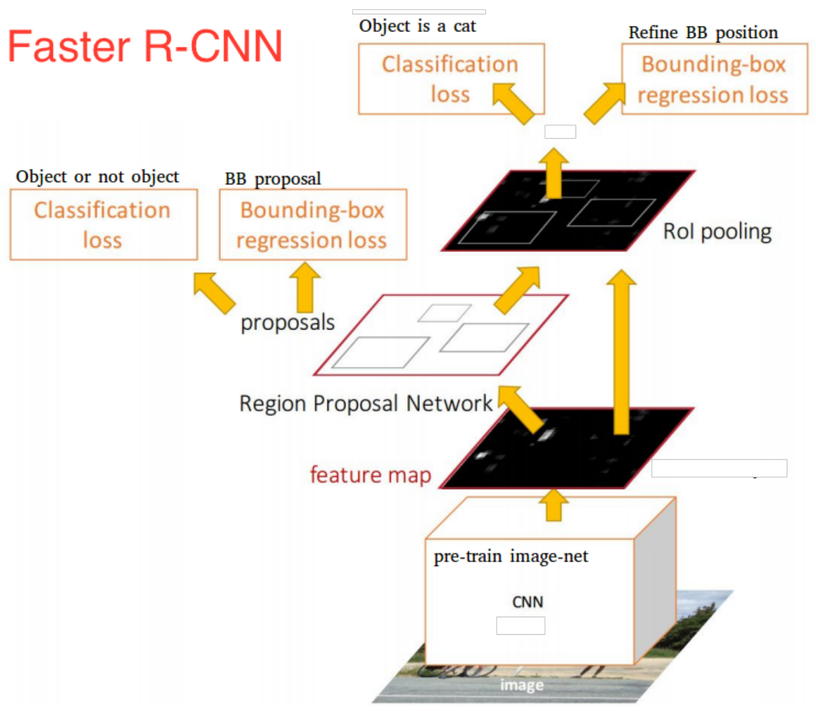


Figure 1.12: Faster R-CNN architecture. Source: [11].

work to merge highly-overlapping bounding boxes of the same object into a single one.

Due to its simplicity, it is much faster than other detection algorithms mentioned. Depending on the backbone architecture, it can run approximately 15 to 150 frames per second (FPS). The limitations of YOLO are that it struggles with small objects and unusual aspect ratios, which is due to the spatial constraints of the algorithm. However, the authors tried to improve this shortcoming in the newly released YOLOv3 with their modified spatial pyramid pooling (SPP) architecture.

1.3.6 Face recognition

Face recognition [13] is a prominent biometric technique that is used for everything from automatically tagging pictures on social networks to unlocking cell phones. Its main components are an object detector trained to detect face regions and classifier that embeds faces into vectors.

It has been a long-standing research topic in the CV community because it uses CV algorithms to extract specific and distinctive information about a person's face, such as distance between the eyes, shape of the chin. This information is then converted to a compact feature vector and stored into a

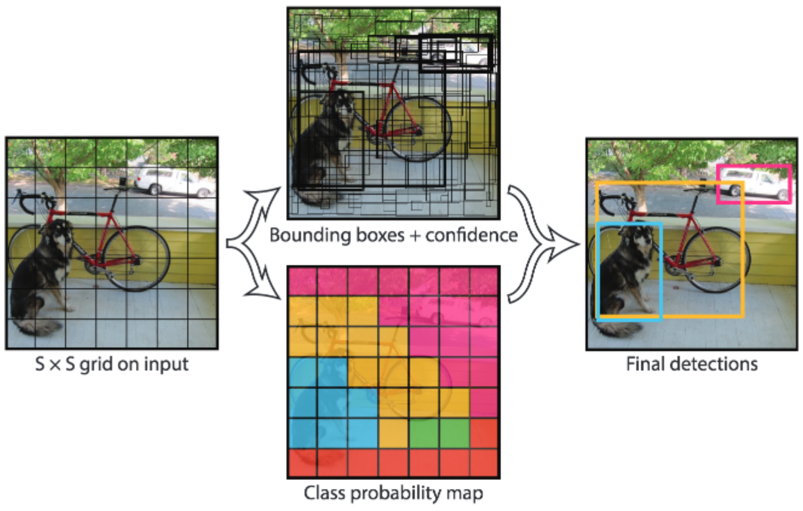


Figure 1.13: Example of YOLO approach. Source: [12].

face database. It is desired only to include specific details that can distinguish one face from another to maintain a reasonable size of the feature vector.

Each facial feature vector can be compared to others to find the most likely match which helps to verify personal identity. However, some face recognition systems are designed to calculate a probability match score to provide several potential matches, instead of just returning a single result. The results of face recognition systems can vary under challenging conditions such as poor lighting, low resolution, improper angle of view.

For decades, only traditional methods such as filtering responses, a histogram of the feature codes, or distribution of the dictionary atoms were used to recognize faces. However, these approaches were improving the accuracy very slowly and were suffering from a lack of distinctiveness and compactness. Effects of lighting, pose and expression drastically worsened the results. Fortunately, this has all changed with the advent of DL [45].

In 2014, DeepFace [93] approach achieved the state-of-the-art accuracy on the famous face recognition benchmark, approaching human performance on the unconstrained condition for the first time. This has caused this field to move in the direction of DL, and it completely reshaped all aspects of face recognition. On Fig. 1.14 the typical pipeline of face recognition is visualized.

1.4 Software frameworks

Research in the fields of AI requires the use of analytical tools, technologies, and languages to help extract insights and value from data, and after that build sustainable prediction model. The key advantages of using frameworks

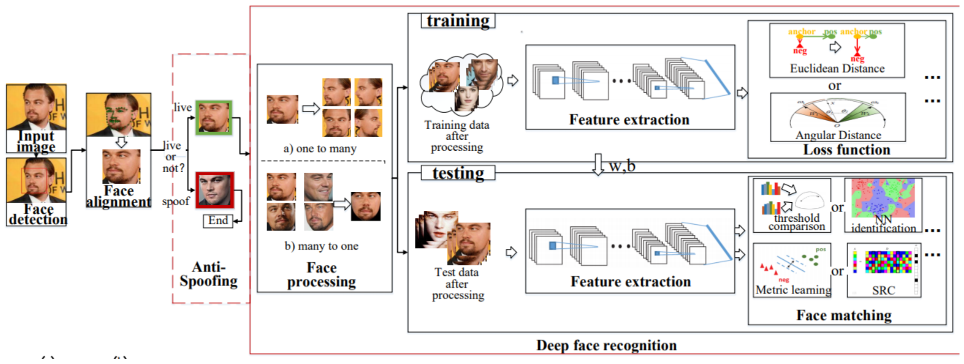


Figure 1.14: Common pipeline for state-of-the-art face recognition systems. Source: [13].

and libraries are the ability to quickly develop and test new ideas and run efficiently on GPUs.

According to a 2017 survey of 16,000 data scientists by Kaggle revealed that researches in AI most rely on Python language [94]. So there is no surprise that most of these packages are developed for that language. For this reason, we provide a quick overview of significant and widespread Python ML and CV frameworks.

1.4.1 OpenCV

OpenCV [95] is a popular, cross-platform, and open-source CV and ML library written in C/C++ language. It was originally sponsored by Intel but now driven mostly by the open-source community with more than 47 thousand people. It has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, extract 3D models of objects, stitch images together, etc. OpenCV is used extensively in companies, research groups, and governmental bodies.

1.4.2 Darknet

Darknet [96] is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. It is the basis for YOLO-based architectures which are one of the fastest object detection CNNs ever proposed.

1.4.3 TensorFlow

TensorFlow [97] is an open source software library for numerical computation using data flow graphs. The graph nodes represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture enables users to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. TensorFlow also includes TensorBoard, a data visualization toolkit.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.

TensorFlow provides stable Python and C APIs as well as non-guaranteed backward compatible API's for C++, Go, Java, JavaScript, and Swift.

1.4.4 PyTorch

PyTorch [98] is an open source library designed to enable rapid research on machine learning models. It builds upon a few projects, most notably Lua Torch, Chainer, and HIPS Autograd, and provides a high-performance environment with easy access to automatic differentiation of models executed on different devices (CPU and GPU). To make prototyping easier, PyTorch does not follow the symbolic approach used in many other deep learning frameworks, but focuses on differentiation of purely imperative programs, with a focus on extensibility and low overhead.

1.4.5 Caffe2

Caffe2 [99] aims to provide a smooth and straightforward way for users to experiment with deep learning and leverage community contributions of new models and algorithms. Users can bring their creations to scale using the power of GPUs in the cloud or to the masses on mobile with Caffe2's cross-platform libraries. In May 2018, the development team decided to merge Caffe2 into PyTorch and make them a single package to enable a smooth transition from fast prototyping to fast execution.

1.4.6 Theano

Theano [100] is a Python library that allows defining, optimizing, and evaluating mathematical expressions involving multi-dimensional arrays efficiently. Since its introduction in 2008, it has been one of the most used CPU and GPU mathematical compilers – especially in the machine learning community – and has shown steady performance improvements. However, the development team decided to stop further releases in 2017.

1.5 Evaluation metrics

In this section, we provide a brief overview of standard evaluation metrics relevant to the thesis tasks.

1.5.1 Intersection over union (IoU)

Intersection over union (IoU) [101], also known as the Jaccard index, is an evaluation metric used as a similarity measure in any object detection task. It measures the ratio between predicted and ground-truth bounding box given by:

$$\text{IoU} = \frac{\text{Area of overlap}}{\text{Area of union}} \tag{1.9}$$

The possible values for IoU are from $[0, 1]$ and their significance can be found in Fig 1.15.

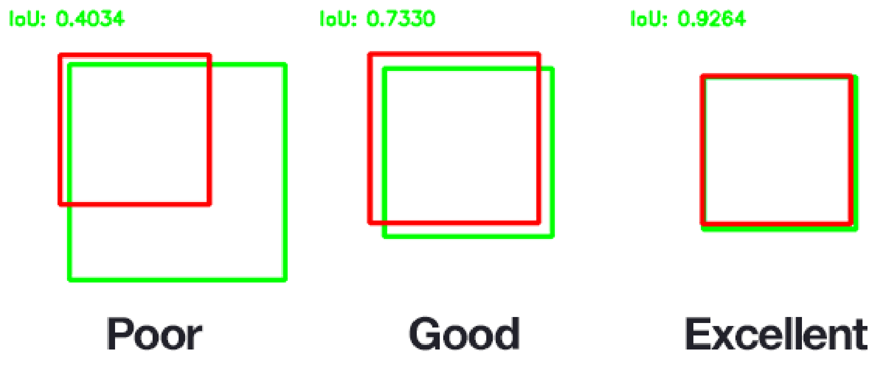


Figure 1.15: An example of computing Intersection over Unions for various bounding boxes. The predicted bounding boxes are drawn in red color, while the ground-truth ones are drawn in green color. Source: [14].

1.5.2 Multiple object tracking accuracy (MOTA)

The multiple object tracking accuracy (MOTA) [2] is the most widely used metric for evaluating a tracker's performance. Its main advantage is expressiveness as it combines three different sources of errors:

- **False positives (FP)** – The number of incorrect detections.
- **False negatives (FN)** – The number of missed detections.
- **Identity switches (IDSW)** – The number of identity switches.

The MOTA equation is then given by:

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDSW}_t)}{\sum_t \text{GT}_t}, \tag{1.10}$$

where t is the frame index, and GT is the number of ground-truth objects. The possible values are reported in the percentage from $(-\infty, 100]$. Negative values are possible when the number of errors made exceeds the number of all objects in sequence.

1.5.3 Multiple object tracking precision (MOTP)

The multiple object tracking precision (MOTP) [2] represents the average dissimilarity between the ground-truth and the predicted bounding boxes. It is calculated as

$$\text{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}, \tag{1.11}$$

where c_t denotes the number of matches in frame t and $d_{t,i}$ denotes the IoU value of ground-truth and the predicted target. It thereby gives the average overlap between all correctly matched hypotheses and their respective objects. [2]

1.5.4 Mean absolute error (MAE)

The mean absolute error (MAE) [102] is a metric used to measure accuracy for continuous variables. It is the average error over all test samples where all individual samples have equal weight. It is given by

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}, \tag{1.12}$$

where n is the number of samples, y_i is the ground-truth value of the i-th sample, and x_i is predicted the value of the i-th sample.

1.5.5 Frames per second (FPS)

The FPS is a common metric used in many fields to measure the run-time performance of camera equipment, algorithms, animations, rendering, etc. In this context, it expresses the ability of an algorithm to process and analyze a certain number of images per second. With increasing algorithm complexity, the frame rate decreases. Real-time ability in video sequence processing is assumed at 25 FPS [2].

Chapter 2

Related work

In this chapter, we briefly present relevant work addressing similar problems in MOT and soft-biometrics field. Given the vastness of the topic, we only limit the review to significant work.

2.1 Multiple object tracking

Most state-of-the-art algorithms addressing MOT follow the tracking-by-detection approach, which heavily relies on the performance of the underlying object detector. However, the trend is now shifting to the end-to-end learning solutions and constructing stronger similarity scores based on appearance, motion, and interaction cues. Although recent NN based detectors have outperformed all other methods for object detection [52, 92, 12], MOT remains a challenging and popular topic.

Simple online and real-time tracking (SORT) [29] is the first pragmatic approach where the main focus is to associate objects efficiently for online and real-time applications. They showed that the quality of detections plays a crucial role in tracking performance and according to their experiments, they can improve tracking by almost 20 %, depending on the detector. Despite using an only simple combination of standard techniques as the Kalman Filter for motion prediction and Hungarian algorithm for the association of the tracks, they were able to achieve comparable performance to other state-of-the-art online trackers.

By adding a deep association metric to SORT [103], it was successfully integrated with the appearance model that improves the tracking performance. The appearance model is based on CNN trained on large scale person ReID dataset. Due to this extension, the algorithm can track objects through longer periods of occlusion, effectively reducing the number of identity switches. The framework of this paper is used for our tracking task. Thus it is more explained in the next chapter.

In 2016, revolutionary end-to-end learning approach based on recurrent neural network (RNN) [48] has been introduced in a novel [104]. Their proposed long short-term memory (LSTM) [105] architecture is capable of performing all multi-target tracking tasks including prediction, data association, state update as well as initiation and termination of targets within a unified network structure. One of the main advantages of this approach is that it is completely model-free, i.e., it does not require any prior knowledge about target dynamics, clutter distributions, etc. However, the object detections should be given as input.

It was not the only case of successful use of RNN. In the paper [106] published year later, a new approach combining multiple cues such as appearance, movement, and interaction is proposed. These cues are fed into LSTM [105] architecture, which learns and remembers the dependencies in a sequence of observation, in contrast to pairwise similarity where only the observations from the current and previous frames are used. Their proposed framework follows end-to-end fashion, and their architecture does not require object detections as input.

Competitive tracking results can be achieved even without sophisticated tracking methods. Tractor [30] accomplishes tracking without following the common tracking-by-detection approach and authors performed no training or optimization on tracking data. Instead, they exploited the bounding box regression of an object detector to perform temporal realignment and to predict the position of an object in the next frame. They also provide a simple extension to this approach, in the form of Siamese NN for ReID and motion analysis model, which achieve state-of-the-art performance on tracking benchmarks.

Results of recent tracking evaluations show that bounding box tracking performance is saturating [21]. Further improvements will only be possible when moving to the pixel level., which is a reason why the authors of recent 2019 work [107] are expanding from MOT to multiple object tracking and segmentation (MOTS). They propose new TrackR-CNN baseline method which jointly addresses detection, tracking, and segmentation with a single convolutional network that extends Mask R-CNN architecture with 3D convolutions to incorporate temporal information, and by an association head which is used to link object identities over time. They also provide evaluation metrics and new dataset with masks for over one thousand distinct objects in ten thousand frames. The main advantage of MOTS is that segmentation based tracking results, are by definition non-overlapping and can thus be compared to ground truth straightforwardly.

2.2 Soft-biometrics

There has not been much written about the general use of biometrics in retail yet. However, some exceptions exist. For example, in [108] authors propose semi-supervised intelligent multi-camera surveillance framework that can perform multiple tasks, including camera management, camera calibration, and multi-view object tracking with ReID based on appearance soft-biometric traits. The survey [109] from 2016 provides a complete overview of innovative camera approaches that can be applied in the retail environment. The study [110] aims at recognizing the age, gender, presence of eyeglasses and beard of passers-by in a retail store. Their solution lies in custom CNN architecture that extracts these traits.

Although end-to-end solution targeting solely retail environment have not been proposed yet, there are plenty of works that deal only with certain soft-biometric features.

2.2.1 Body traits

Frequent body traits [110] extracted from an RGB camera are height, gait, and color of clothes. The best feature for distinctiveness between individuals proved to be gait, which is the reason why it is often used in forensic analysis. However, there are not many uses for gait in the retail environment. Thus we only focus on others.

2.2.1.1 Height estimation

There has been a long history in determining an individual's height. One of the essential works in this field is [19], where authors proved that height estimation is possible without any information about camera parameters, only several scene correspondences with known real-world measurements are sufficient. In their work, they also describe how the affine 3D geometry of a scene can be reconstructed from a single image.

In [111] authors describe a simple uncalibrated model of error distribution in height as a function of the location of the object in the image and the estimated camera height. Their approach builds on previous work [19] and improves the accuracy in estimating the height while reducing the burden of reliably computing the ground plane.

More recent work [112] uses a single calibrated camera, more explicitly assuming the knowledge about its pose concerning the world and vanishing point in the reference direction. According to their proposed theorem, by knowing the proportion of camera height concerning the object height, at the object's position in the image plane, it is possible to estimate the height of the object in the real world. Their presented method gives accurate results in unstructured environments, regardless of the relative distance from the camera.

Authors of [113] proposed a framework utilizing the calibrated camera for estimating height in video surveillance. Their primary assumption is that it is possible to obtain a camera's focal length, tilting angle, and height by using non-linear regression model from the observer's head and foot points of people in the scene, instead of estimating the vanishing point and vanishing line. Once these calibration parameters of a camera are obtained, the physical height of a person can be estimated from a pair of head and foot points observed from the image and their proposed formula.

2.2.1.2 Color clothes estimation

One of the essential works [114] in this field comes from 2012, where authors focus on fashion photographs and propose an effective method to produce intricate and accurate parse of a person's outfit. They also introduced large labeled dataset with available labeling tools. Finally, they designed a prototype application for visual garment search.

DeepFashion [115] is another large-scale clothes dataset with extensive annotations introduced in 2016. Authors also introduced a new baseline method, namely FashionNet, which learns clothing features by jointly predicting clothing attributes and landmarks. The estimated landmarks are then employed to pool or gate the learned features.

The current state-of-the-art is the baseline model built upon Mask R-CNN [116], termed Match R-CNN [117]. The authors of this work also proposed new challenging large-scale dataset DeepFashion2. Their presented model offers end-to-end training framework that jointly learns clothes detection, landmark estimation, instance segmentation, and consumer-to-shop retrieval.

2.2.2 Facial traits

If we focus only on the head of individuals, then many algorithms targeting facial biometric [13] are regularly introduced. Most of the work from this field comes inevitably from face recognition task, which is natural since this field has a history almost as long as fingerprint recognition. Commonly extracted facial soft-biometrics are gender, ethnicity, age, emotion, pose, glasses, beard, and mustache with a frequent goal to explore their discrimination capabilities among other individuals during in-the-wild scenarios. We will limit this review to only the most important ones that can be used for retail use, namely age, gender, emotion, and pose [118]. Some of these traits are commonly estimated together by using one model.

2.2.2.1 Multi-task approaches

In paper [119], authors propose multi-task CNN for face recognition where identity classification is the main task and pose, illumination, and expression estimations are the side tasks. They also provide analysis of multi-task learning with extensive experiments that demonstrate the effectiveness of the proposed approach.

HyperFace [120] is an algorithm that can simultaneously do face detection, landmarks localization, pose estimation and gender recognition. It uses CNN model architecture that fuses intermediate layers of a CNN using a separate CNN followed by a multi-task learning algorithm that operates on the fused features. It exploits the synergy among the tasks which boosts up their performances. They also proposed two variants with a different focus – HyperFace-ResNet that achieves high accuracy and Fast-HyperFace that significantly improves the speed of the algorithm. Their experiments show that the proposed models can capture both global and local information in faces and perform significantly better than many competitive algorithms for each of these four tasks.

The authors of the same composition later released improved version [121] that can additionally solve for tasks of smile detection, age estimation, and face recognition. They also extended their approach by training the model on multiple datasets whereas HyperFace was trained only on one.

2.2.2.2 Age and gender estimation

In most retail cases, we are interested in estimating age at the same time as gender [121, 122], since these traits play a crucial role in targeted advertising. In the case of age, it is very challenging to obtain it by the camera because the apparent age sometimes significantly differ from the real one. Fortunately, particular numbers are not that important, but instead categories such as child, youth, adult, middle-age and elderly. On the other hand with gender, the situation is slightly simpler because there are several other attributes that algorithms can use. We further present only recent proposals from this field that mostly relies on DL models.

The first ever work applying DL methods was presented by the authors in [122]. They propose a CNN-based framework for age estimation, and instead of using only the feature map obtained at the top layer, they utilize feature maps obtained in different layers for the final estimation. Additionally, they incorporate a manifold learning algorithm in the proposed scheme, and that significantly improves the performance.

In the same year, work [123] introduced Deep EXpectation (DEX) method that tackles the estimation of apparent age in face images. They proposed CNN with VGG-16 architecture pre-trained on ImageNet. They also introduced large-scale face images dataset with available age to pre-train their CNN.

Recently in 2019, authors of [124] proposed Soft Stagewise Regression Network (SSR-Net), a light-weight CNN model with real-time performance that is targeting age and gender estimation. Their work is inspired by DEX, and they address age estimation by performing multi-class classification and then turning classification results into regression by calculating the expected values. Their greatest benefit is the compactness of the model presented. SSR-Net takes only 0.32 MB of memory, and despite its size, it is approaching the performance of more than 1500× larger state-of-the-art methods.

2.2.2.3 Facial expression recognition

Facial expression [125] is one of the main clues that can be utilized in retail stores to measure the satisfaction of the consumers. [126] is a very popular paper that proposes a general CNN framework with real-time performance. Their model is capable of accomplishing the tasks of face detection, gender classification, and emotion classification simultaneously in one blended step using the proposed architecture. They validate their results on recent datasets, but also by building their robot which succeeded in the competition. Their architecture and pre-trained models are released under an open-source license.

The framework known as EmoPy [127] is a toolkit with multiple implemented CNN architectures for emotion recognition. For example, they use time-delayed 3-D CNN that utilizes temporal information as part of its training samples. In other words, instead of using an only single image for prediction, it uses past images from a series for additional context. The idea is to capture the progression of a facial expression leading up to a peak emotion. They also propose hybrid LSTM [105] architecture that combines approaches from CNN and RNN to include temporal context from whole still images, not only from face patches as the former presented. They also introduce a simpler model trained on FER2013 dataset [125] which can make predictions based on a single image. Moreover, in their framework, it is possible to specify only a smaller subset of all seven emotions available, which is undoubtedly more practical in real-world scenarios.

Paper [128] from 2018 has presented a novel approach for facial expression recognition using deep sparse autoencoders (DSAE) [129], which can automatically distinguish the expressions with high accuracy. Both the facial geometric and appearance features have been introduced to compose a high-dimensional feature with accurate and comprehensive information of emotions. In the end, the experiment results have demonstrated that the proposed approach outperforms the other three state-of-the-art methods by a large margin.

2.2.2.4 Head pose estimation

Head pose estimation [13] is closely related to other facial analysis problems, and it might be useful in a retail store when there is a requirement for modeling customer attention on specific products or areas. Traditional algorithms work by estimating facial landmarks (key points) from the target face and then solving the 2D to 3D correspondence problem with a mean human head model as a reference. Since it is a very fragile method that relies on landmark detection performance, current state-of-the-art methods focus on detecting the pose without detecting the key points.

In 2018, an accurate and easy to use head pose estimation CNN known as Hopenet [130] was introduced. In their work, authors can detect intrinsic Euler angles (yaw, pitch, and roll) directly from image intensities through joint binned pose classification and regression. They showed their network generalization capacity by testing the performance on various external datasets without fine-tuning.

A novel method is known as FSA-Net [131] also targets head pose estimation. Authors of this work proposed a light-weight non-landmark CNN model that even outperforms methods utilizing multi-modality information from depth and RGB cameras.

Chapter 3

Methodology

This chapter provides an overview of our designed procedures and algorithms. As mentioned in the earlier in the thesis objectives, the goal is to design and implement a framework that tracks people in the video sequence followed by additional information extraction on person-specific features. Before getting into our the tracker design, we present our data acquisition and data pre-processing process. Later then, we propose a people detector and tracker which follows the well-known tracking-by-detection approach and in the end, we present strategies for additional information extraction.

3.1 Image acquisition

Image acquisition is the first stage in any vision processing system. The goal of image acquisition is to transform real-world optical data into an array of numbers that can be processed by a computer. The process heavily depends on the camera and lens hardware.

The main component of the camera is its sensor that converts light into electrical charges, while the lens focuses the light on particular spots of the sensor. Nowadays, the most used technology for the camera sensor is complementary metal oxide semiconductor (CMOS) due to its low noise and processing speed capabilities. Since the work is done in cooperation with well-equipped Image Processing Laboratory (ImproLab) at Faculty of Information Technology (FIT) Czech Technical University (CTU), we have a choice of several camera and lens variants available.

Based on assumptions and thesis instructions, we know the scene, and its parameters in advance. To test the designed algorithms of this work, we chose the scene to be the 14th floor of the Faculty of Civil Engineering (FCE) CTU building near elevators. Based on the scene, we selected the appropriate combination of lens and camera sensitive to the visible spectrum. The ImproLab has most of its cameras from Basler manufacturer. Thus we decided to utilize Basler ace acA1920-50gc [15], the Gigabit Ethernet (GigE) camera with the Sony IMX174 CMOS sensor that delivers 50 FPS at 2.3 megapixel (MP) resolution. Due to the spatial properties of the scene, we chose Basler 8mm lens C125-0818-5M F1.8 [16] that proved to be the best option in this case.

(a)

(b)

Figure 3.1: Selected camera equipment for our task. a) Camera Basler ace acA1920-50gc. b) Lens Basler 8mm C125-0818-5M F1.8. Source: [15, 16]

.

The image stream from Basler's camera equipment can be obtained in several ways. The most convenient but limiting approach is to use Basler pylon Camera Software Suite, which is a software package comprised of an easy-to-use software development kit (SDK), drivers and tools that can be used to operate any Basler camera using the user interface on multiple platforms such as Windows, Linux, and Mac. However, this option does not allow integration with custom software. Another option is to use Basler's pypylon [132] which is the new open source wrapper interface for the Basler pylon Camera Software Suite that allows direct camera access through Python code.

Due to the nature of the work, the data can be in an offline form. Thus we decided to use the first method with the Basler pylon framework because it is more convenient than the pypylon wrapper.

We record the video sequence at the highest possible resolution, i.e., 1920x1080 and at 25 FPS. Each frame is saved in a jpg format to avoid an overhead that would be caused by decoding the video formats. The camera shutter speed is set manually so that people in the scene are not motion blurred and the aperture is set to the sweet-spot of the lens, which in our case is f/4.0, to achieve maximum image quality. Since these parameters would make the image too dark, it is necessary to increase the sensor gain value. In this case, we decided to set sensitivity value to automatic, which means that it is always calculated so that the scene is exposed correctly. We set the white balance to a fixed value determined by automatic detection to avoid drastic color changes during shooting. We provide an example of the scene in Fig. 3.2.

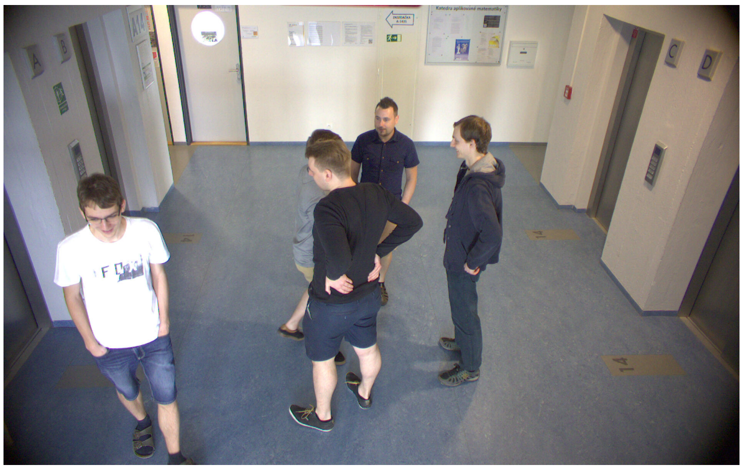


Figure 3.2: Example of the acquired image and the scene layout. We can see that despite being a quality lens, it contains distortion and vignetting.

3.2 Image pre-processing

After the images are obtained, various pre-processing methods can be applied to improve the accuracy of feature extraction and classification methods. However, in our case, we assume that all images are acquired in satisfactory quality and with proper exposure, shutter speed, and white balance settings. The only pre-processing we do is removing lens distortion by using a camera calibration method as described in 1.3.1.

3.3 Detection and tracking components

In this section, we present our design decisions for two core components of our framework – the detector and the tracker. In contrast to batch based approaches, we primarily target online tracking where only information from the previous and current frame is available. We also try to maintain both components as much as efficient as possible for further extension to real-time applications.

The MOT can be viewed as a data association problem, where the goal is to associate hypotheses obtained from the object detector across frames in a video sequence. One way to solve this problem is to compare and match specific features such as the appearance and motion of objects in the scene.

The methods employed in the tracker are inspired by work [100], that performs Kalman filtering [76] in image space and frame-by-frame data association using the Hungarian method [133] with an association metric that measures bounding box overlap, and by [103] that improves the previous approach by adding deep association metric.

3.3.1 People detection

The beginning of the tracking process is an object detection step. In this work, we use various detector models based on YOLO and Faster R-CNN with ResNet50 backbone. The result of forwarding the frame through CNN architecture is in the form of hypotheses that contains bounding box coordinates, object class, and detection confidence. Except for input image resizing to be compatible with the CNN, we do not perform any other pre-processing in this phase.

Object detectors are usually trained on multiple object classes, but as we are only interested in people, we filter out all other classes from the output hypotheses. We also filter out low-confidence ones. The confidence threshold is variably configured according to the object detector used. Faster R-CNN produces a lot of false positives. Thus we take into account only hypotheses with confidence greater than 90 %. In YOLO, the threshold value of 70 % has been experimentally verified to work well. We employ both network architectures with their default parameters trained on the COCO dataset [134]. A common problem of object detectors is that they often generate more overlapping hypotheses than the ground-truth objects (false positives). To handle the removal of overlapping bounding boxes we use additional NMS step [17] for bounding box post-processing.

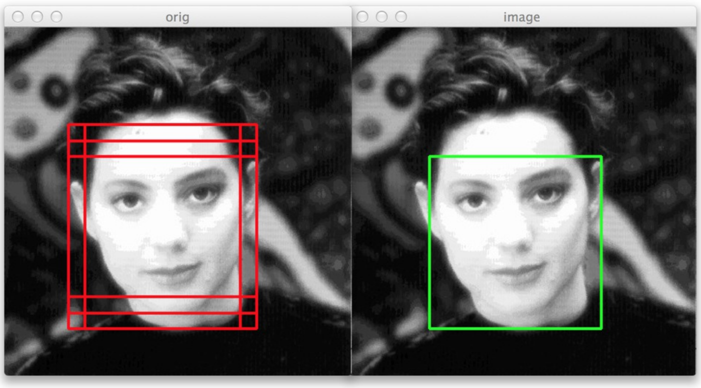


Figure 3.3: Six bounding boxes are detected around the face, but by applying non-maximum suppression, it is possible to remove the redundant ones. Source: [17].

3.3.2 Track handling

To maintain object identities and propagate object states into the future, we need to maintain historical information in the memory about previous detections. To achieve this goal, we define track object class, and in each frame, we associate current detections with existing track objects (tracks). While the detected hypothesis is only represented by an object class, bounding box and confidence, a track contains multiple attributes such as identity, status, time since the last update, age in frame units, bounding box, state, and class-specific information.

When a person enters or leaves the scene, unique identity needs to be created or removed accordingly. For initiating a new track, we take detections that were not successfully associated in the matching phase of the current frame (unmatched detections). These fresh tracks are then set to tentative during their first t_c frames. Tracks that are not successfully associated during the t_c frames are removed from the scene, which helps to accumulate enough evidence to prevent tracking of false targets. Each track also has a variable that determines the number of frames of the last successful association. Tracks are terminated if they are not associated within t_m frames or their state predictions are outside of the frame. Based on our experiments we have selected a value $t_c = 7$ for track confirmation and $t_m = 50$ for track termination. Due to camera settings, the termination based on our value of t_m means that the appropriate detection was not found for two seconds.

3.3.3 State estimation

The state representation helps to propagate a track's identity into the next frames. The inter-frame displacements of each track are approximated with a linear constant velocity model which is independent of other objects. Our design of the track's state is represented as a 10-dimensional vector:

$$x = \left(cx, cy, ty, h, r, \dot{cx}, \dot{cy}, \dot{ty}, \dot{h}, \dot{r}\right), \tag{3.1}$$

where cx, cy represent center pixel location of the bounding box, ty represent the top center location of the bounding box, h is pixel height, and r is the aspect ratio of the bounding box. Each of these parameters has its respective velocities in image coordinates.

To predict and estimate the accurate value of track states, we use standard Kalman filter [76]. It is a recursive algorithm with a real-time performance that is commonly used to extract the useful signal from noisy measurements. In this case, it can be used to refine the bounding box position.

The whole filtering process consists of two main steps – prediction and correction. In the prediction phase, the filter produces predictions of the current state variables, along with their uncertainties. Furthermore, once the output of the next measurement is known, it is used in a correction step to update state variables with a weighted average. The weights are greater for values with higher certainty.

The Kalman filter assumes that the real state at time k can be obtained from state $k-1$ by the following prediction equation:

$$x_k = F_k x_{k-1} + B_k u_k + w_k, \tag{3.2}$$

where F_k is the state-transition model, B_k is the control-input model applied to the control vector u_k, w_k is the process noise, assumed to be drawn from $w_k \sim \mathcal{N}(0, Q_k)$, and Q_k is covariance matrix with appropriate dimensions.

Measurements z_k at time k then can be obtained as follows:

$$z_k = H_k x_k + v_k, \tag{3.3}$$

where H_k is the observation model which maps the true state space into the observed space, v_k is the observation noise, assumed to be drawn from $v_k \sim \mathcal{N}(0, R_k)$, and R_k is covariance matrix with appropriate dimensions. All filter parameters are supposed to be correctly user-defined. Otherwise, the filter will not have the required properties.

In our design, we do not assume the variability of matrices F_k, Q_k, H_k, R_k in time and we can drop B_k since we do not have any control input. F is then designed according to state vector (defined in 3.1) and constant velocity model as follows:

$$F = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & dt & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & dt & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & dt & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & dt & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & dt \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}, \tag{3.4}$$

where dt is chosen to be 1. Matrix H for obtaining measurements is given by:

$$H = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}. \tag{3.5}$$

The noise matrices are heavily dependant on the captured scene, but generally speaking, we set significantly higher variance values in for measurement noise matrix R then process noise matrix Q. This is because an object detector produces sparse detections with noisy bounding box predictions.

To sum it up, each track has its state as defined in 3.1. When detection is associated with a track, the bounding box coordinates are used as a measurement to update the internal state of the filter. Velocity components are then solved optimally via the filter's transition matrix. If detection is not associated with a track, its state is predicted using a linear model (skipping the correction step).

More details on the filter assumptions, properties and calculations can be found in its original proposal [76].

3.3.4 Appearance features

Using only state estimation in the matching phase would result in a high number of identity switches due to occlusion effects and non-linearity in people's behavior. Therefore, appearance descriptors are often extracted to make hypothesis-to-track matching more robust. At the beginning of this work, we tried to use color histograms of bounding box patches as appearance descriptor. Patches were then converted to hue-saturation-value (HSV) color space to make them comparable by Bhattacharyya distance [135] with hue and value channel as input. However, object histograms are greatly affected by their background. Thus this solution was not robust, and we decided not to continue this path.

Instead, we utilized a CNN model from [103] that has been trained to discriminate pedestrians on a large-scale dataset. The input of the model is a bounding box patch of a person and based on that it outputs 128-dimensional vector comparable with cosine distance metric. Through the integration of this feature descriptor, it is possible to recover track's identities even in long-term occlusions.

3.3.5 Data association

To create associations between existing tracks and newly arrived detections it is convenient to solve the assignment problem using the Hungarian algorithm [133]. The values of cost matrix are calculated through a combination of two appropriate metrics – state and appearance.

3.3.5.1 State metric

The cost of the state between track and detection is designed in two variants. The first approach is following the design of [103] that uses squared Mahalanobis distance between track states and detections. Since it includes a covariance matrix, it takes into account estimation uncertainty and is defined as follows:

$$d_m(i,j) = (d_j - y_i)^T S_i^{-1} (d_j - y_i) \, , \tag{3.6}$$

where y_i is i-th track state without velocities, S_i is according to state covariance matrix, and i_j is the j-th bounding box in an appropriate format. The Mahalanobis distance measures how many standard deviations is the detection away from the mean track location. Its value can then be compared and gated against the proper threshold from a cumulative χ^2 distribution.

However, it has proven to be inappropriate for our use based on several experiments. The results often diverged, mostly because we try to approximate non-linear person behavior with a linear filter. Therefore, we decided to use a simpler metric with fewer input parameters. More specifically, the Euclidean distance:

$$d_e(i,j) = \sqrt{(cx_i - cx_j)^2 + (ty_i - ty_j)^2}, \tag{3.7}$$

where cx_k is the x value of center point and ty_k is the y value of top center point of the k-th bounding box. Furthermore, we enhance 3.7 formula so that the output values are in the same order as the outputs of the appearance distance metric:

$$d_{el}(i,j) = log_\alpha \left(\sqrt{(cx_i - cx_j)^2 + (ty_i - ty_j)^2} \right) - 1. \tag{3.8}$$

To suppress possible negative values that would not make sense for distance metrics, we take $\max(d_{el}(i,j), 0)$. Our idea behind this formula is that we want to tolerate small spatial displacements caused by the detector in consecutive frames.

The tolerance amount depends on the base of the logarithm $\alpha \in \mathbb{N}$, which is a hyperparameter that needs to be manually determined based on the task being solved. The influence of α can be best explained by an example. If $d_e(i,j) \leq \alpha$ then $d_{el}(i,j) = 0$, which briefly means that this metric does not penalize bounding boxes that are maximally α pixels distant. The logarithm function then helps to squash the values to lower order. For our type of captured scene, $\alpha = 80$ pixels proved to be working well.

For the sake of clarity, we provide plot in Fig. 3.4 of this distance function with $\alpha = 80$.

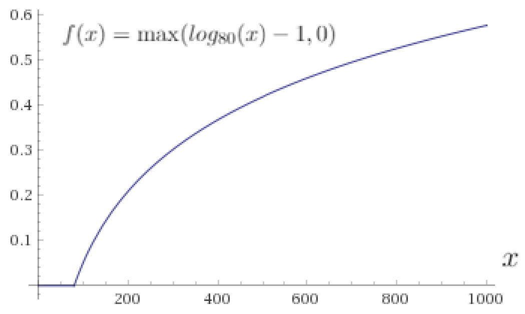


Figure 3.4: Example plot of the distance metric values used for track-to-detection spatial displacement cost.

3.3.5.2 Appearance metric

The appearance metric is used to describe how much are the individuals visually similar, thus helps in distinguishing them. For each detected bounding box patch we calculate its appearance feature vector r_i with $||r_i|| = 1$ by a forward pass of the CNN referred in 3.3.4. Moreover, for each registered track j, we keep a gallery $R_j = \left\{ r_j^{(t)} \right\}_{t=1}^{L_j}$ of the last $L_j = 100$ associated appearance features. Then, the appearance distance between the i-th detection and the j-th track is defined as:

$$d_c(i,j) = \min \left\{ 1 - r_i^T r_j^{(t)} \mid r_j^{(t)} \in R_j \right\}. \tag{3.9}$$

The outcome of $d_c(i,j)$ is neatly bounded in $[0,1]$, where 0 means most similar objects. This time, we introduce a threshold t_c to control if an association is possible. Thus any track and detection pair with $d_c(i,j) > t_c$ is so different that it cannot be associated. For our particular scene, $t_c = 0.15$ has proved to be working well.

3.3.5.3 Matching

Building the associations happens in cascade fashion which is taken over from the [103]. The cost value is based on actual state and appearance and is calculated for each possible track and detection pair. We find the best associations by running the Hungarian algorithm [133] that solves the min cost assignment problem. Then, we verify if the association is possible based on the appearance threshold t_c. If yes, we update registered tracks with associated detections. For the rest of unmatched detections, we create new tentative tracks. In Fig. 3.5 we also provide a simplified diagram of one pass through of the algorithm cycle.

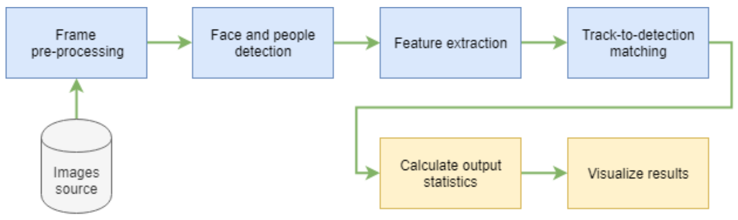

Frame pre-processing
Face and people detection
Feature extraction
Track-to-detection matching
Images source
Calculate output statistics
Visualize results

Figure 3.5: Simplified diagram of single framework pass through. Yellow-marked boxes are optional components used for evaluation.

3.4 Soft-biometrics extraction

Besides the goal of building the people detector and tracker, we also address additional information extraction related to people class – soft-biometrics. The aim is to focus on information that can be utilized in the retail environment. Therefore we have focused on methods for estimating age, gender, emotion, height, and ground plane trajectory. Because extraction all of these features in each frame would be computationally intensive, we design our framework to be fully modular in the sense that any of the feature extractors can be turned on or off at any time.

3.5 Facial features

Before extracting any facial features, robust face detector needs to be utilized to locate faces in the image. Due to the fact the era of traditional methods such as Haar cascades [57] is over, we decided to choose suitable face detector only from CNN-based architectures. In particular, we designed an interface for three different CNN-based face detectors that offer different trade-offs between speed and accuracy performance.

- **faced** [136] is first of them achieving near real-time performance on CPU. It uses an ensemble of two CNNs where first is based on YOLO architecture. It takes the input image and outputs a grid where each cell contains a prediction of bounding boxes and the probability of one face. Furthermore, these outputs are fine-tuned by another custom CNN architecture that is trained to refine bounding box coordinates. Our utilized model is pre-trained on WIDER FACE [137] dataset.

- **Tensorflow Mobilenet SSD** [138] is a popular open-source face detector framework. It is based on a more robust single-shot detector (SSD) [139] architecture than the faced framework, thus it provides better accuracy. We utilize a pre-trained model on WIDER FACE [137] dataset that can have a real-time performance on GPU and takes only about 400 MB of GPU memory.

- **MTCNN** architecture [140] and its implementation [141] is the last utilized face detector that achieves superior accuracy over the state-of-the-art techniques on the public datasets. It leverages a cascaded architecture with three stages of designed CNNs to predict face and landmark location in a coarse-to-fine manner. In addition to face detection, it also performs a face alignment. The proposed architecture is a heavy, but it achieves near real-time performance on modern GPUs. We use this architecture to compare accuracy performance with weaker face detectors.

Gathered face regions from the face detector need to be paired with people detections. This problem is solved based on the assumption that the face is in the upper middle part of the person bounding box. Face regions with confidence less than 90 % or without associated person detection are not preserved. No additional post-processing of face regions is done in this step. With the face regions paired to people, we can finally estimate a few types of soft-biometrics.

3.5.1 Age and gender estimation

As already mentioned in 2.2.2.2, age is most commonly estimated at the same time with gender. In this work, we built an interface for two existing CNN models – Keras Age and Gender [142] based on DEX architecture and SSR-Net [124]. However, in our specific task, the SSR-Net architecture with the pre-trained model provided not only more accurate predictions but better speed performance. Thus we consider only the SSR-Net in the final design.

Before feeding the face image patch into CNN, we first extend the detected face region by margin m as follows:

$$x_{tl\_new} = \max(x_{tl\_old} - m \cdot fa_w, 0) \tag{3.10}$$

$$y_{tl\_new} = \max(y_{tl\_old} - m \cdot fa_h, 0) \tag{3.11}$$

$$x_{br\_new} = \max(x_{br\_old} + m \cdot fa_w, fr_w) \tag{3.12}$$

$$y_{br\_new} = \max(y_{br\_old} + m \cdot fa_h, fr_h), \tag{3.13}$$

where x_{tl_new}, y_{tl_new} and x_{br_new}, y_{br_new} are new top left and bottom right coordinates of bounding box, respectively, $x_{tl\_old}, y_{tl\_old}, x_{br\_old}, y_{br\_old}$ are their corresponding original coordinates, fa_w and fa_h is according width and height of face region, fr_w and fr_h is according width and height of frame.

If the margin m is too small, then the face is not cropped entirely, and if the offsets are too big, then too much space around the face is included. Both cases affect the prediction accuracy, thus it is necessary to choose a suitable margin for a particular scene. In our case, the value margin value of $m = 0.5$ worked well.

Furthermore, face patches are also normalized by min-max normalization from 0 to 255 to eliminate the illumination variance:

$$x_{norm} = \frac{(x - \min(region)) \cdot (newmax - newmin)}{\max(region) - \min(region)} + newmin, \tag{3.14}$$

where x is the input pixel value, $newmin = 0$, $newmax = 255$, and $\min(region)$ and $\max(region)$ takes accordingly minimum or maximum value of the face region.

The output of the age classifier is a continuous value which is not further post-processed. Output values from the gender model are $x \in [0, 1]$ where $x > 0.5$ means female and otherwise male. The more the value of x is at the edge of the interval, the more certain the prediction is. Also, no further post-processing is needed for gender values.

3.5.2 Facial expression recognition

In the same spirit as the previous models, we tried to utilize at least two different architectures to estimate emotions which can be further compared. First of them is [126] inspired by [123] CNN architecture, which introduces the open-source framework that can recognize seven types of face emotions – angry, disgust, fear, happy, sad, surprise, neutral. Their model can operate in real-time, and they provide pre-trained weights on FER2013 dataset [125].

Other architecture employed, known as EmoPy [127], is a toolkit with various CNN architectures implemented. However, the most popular model is Fermodel which is trained to predict the same facial categories as the previously mentioned model. We decided to utilize EmoPy for the task of estimating emotions because their framework can operate with a smaller subset of all seven emotions, which is undoubtedly more practical in real-world scenarios. Furthermore, they provide more complex models that provide an output based on temporal information of past frames. The provided model can operate in real-time and is also pre-trained on FER2013 dataset [125].

For the input of EmoPy, we similarly pre-process the input face region as in 3.5.1. We expand the face region by margin m, but we also convert the image to grayscale. This time, we do not normalize the input image, because the model was not trained in this manner. The output of the model is facial categories with their associated probabilities. We choose the most likely one and ignore the rest.

3.6 Spatial information

In addition to visual information such as facial features, we can extract additional information which relates to the spatial features of the space captured. However, these methods usually need some knowledge of the scene captured, so they cannot be applied in general. However, in our assumptions, we expected the full knowledge of the scene, thus we are not limited, and we can apply a wide range of available methods.

3.6.1 Trajectory

One of the most intuitive ways to get some spatial information in the scene is to focus on the movement in space itself. More specifically, extraction of individual's trajectories. The best way to analyze motion in space and to work with trajectories is in the top-down (or bird's eye) view. However, in most real cases of tracking people, we are unable to place the camera to achieve this kind of perspective. This placement would also prevent the possibility of obtaining further facial descriptors. Fortunately, we can transform some parts of the original image to resemble similar to a top-down view.

Formally, we can utilize 2-D projective transformation between real-world and image ground plane with its main component known as homography matrix. Since the perspective problem is considered as one of the most difficult and precise topics in CV that goes beyond the possibilities of this work, we recommend one of the most popular books [18] in this field, where a detailed explanation can be found. However, if we resort to a brief explanation, then the homography matrix H relates the transformation between two planes (up to a non-zero scale factor s) as follows:

$$s\begin{pmatrix}x'\\y'\\1\end{pmatrix}=H\begin{pmatrix}x\\y\\1\end{pmatrix}=\begin{pmatrix}h_{1,1}&h_{1,2}&h_{1,3}\\h_{2,1}&h_{2,2}&h_{2,3}\\h_{3,1}&h_{3,2}&h_{3,3}\end{pmatrix}\begin{pmatrix}x\\y\\1\end{pmatrix}. \tag{3.15}$$

H has eight degrees of freedom, which means there are eight unknowns that need to be solved for, and it is generally normalized with $h_{3,3}=1$ or $h_{1,1}^2+h_{1,2}^2+h_{1,3}^2+h_{2,1}^2+h_{2,2}^2+h_{2,3}^2+h_{3,1}^2+h_{3,2}^2+h_{3,3}^2=1$.

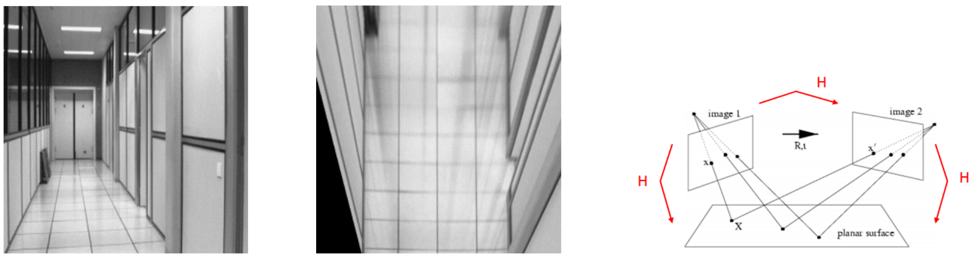
Typically, an estimate of the matrix H is done by finding point correspondences between the target planes. One point correspondence gives two linearly independent equations, and four correspondences are needed for a minimal solution. If more than four correspondences are known, then a more accurate solution can found according to the predefined cost function. Finally, the matrix H can be estimated running Direct Linear Transform (DLT) algorithm [18].

There are many scenarios in CV where a homography may be required. However, the question is how it relates to trajectories. If we have at least four ground-truth world measurements of the ground plane that are visible in the camera frame, we can build frame-to-world point correspondences and estimate the matrix H. Further, we can use this matrix to transform each point (x, y) that lies on the ground plane in scene coordinates to real-world coordinates. For a specific example of tracking people, we assume that people are walking on a common ground plane. Thus we can take their bottom center point of the bounding box and transform it with H to get the corresponding point in real-world coordinates. As a result, we can precisely estimate positions where people have walked, not in frame coordinates, but in real-world coordinates that we previously measured in the calibration phase.

3.6.2 Height estimation

In the previous section, we presented a 2-D transformation that can transform point coordinates between planes. This section deals with a related problem known as single-view metrology, which uses projective geometry to recover a structure of the 3-D world objects from a single image with several reference measurements. Since this section again contains complex theoretical concepts such as vanishing points, vanishing lines, we assume that the reader got acquainted with them from the suggested book [18].

Let us have a visible ground plane p represented as a floor in the scene, the vanishing point v in the vertical direction of the p, vanishing line l of the p, t_r and b_r are the top and base points of the reference object, respectively and t_x and b_x are the top and base points of the object to be measured. All base points must lie on the p, and all top points must lie on a line that intersects



(a) (b) (c)

Figure 3.6: Example of the 2-D projective transformation. a) Source image. b) Top-down view of the corridor floor generated from (a) using the four corners of a floor tile to compute the homography. c) A planar surface viewed by two camera positions is related by homography H, thus any point can be transformed between these different planes. Source: [18]

.

the according to the base point and is perpendicular to p. Then, the α metric factor can be found by the following equation:

$$\alpha = -\frac{||b_r \times t_r||}{Z_r(l \cdot b_r)||v \times t_r||}. \tag{3.16}$$

With the α calculated, we can estimate a height Z_x for an arbitrary object that stands on the ground plane by the following equation:

$$Z_x = -\frac{||b_x \times t_x||}{\alpha(l \cdot b_x)||v \times t_x||}. \tag{3.17}$$

The application of this formula is most easily understood from Fig. 3.7 and the proof can be found in [19].

To conclude, with this single-view metrology framework, we can measure the real-world height of the object that stands on the ground plane (with known t_x and b_x that are bounding box top or bottom point, respectively). It works for uncalibrated cameras, only real-world measurements on the ground plane and beyond are sufficient for calculating the v and l.

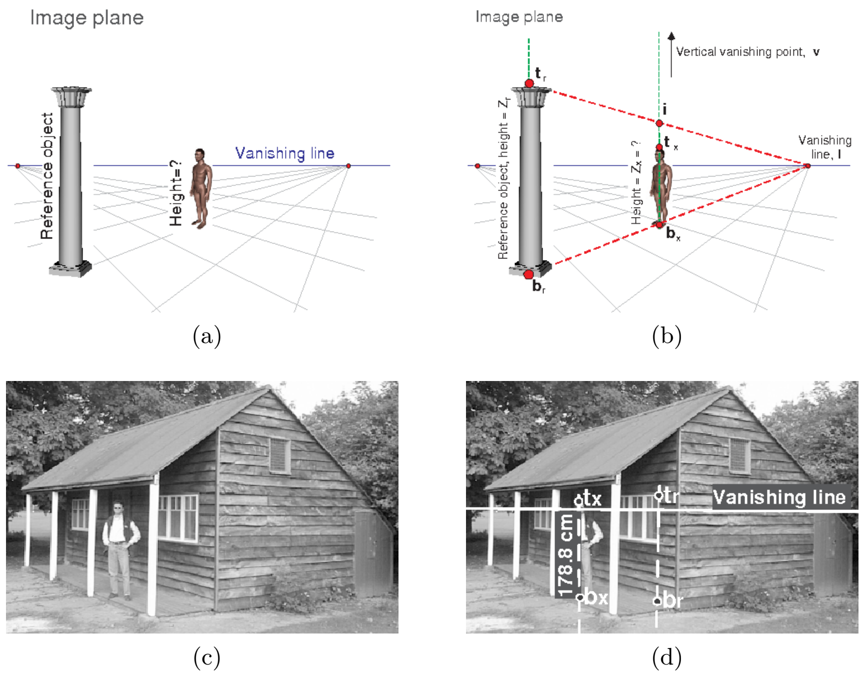


Figure 3.7: Measuring heights in single images. (a) The aim is to compute the height of the human figure relative to the height of the column (reference). The vanishing line of the ground plane has been computed and is shown blue. (b) The unknown height Z_x can be computed from image quantities only as shown in 3.17 (c) A photograph of a garden shed in Oxford. (d) Once the height of the window top edge from the floor has been measured (reference), the height of the man is computed to be 178.8 cm which is about 1 cm off the ground truth. Source: [19]

.

Chapter 4

Implementation

This chapter contains information related to the implementation of the proposed framework. The whole implementation is done in Python language with the dependence on several other frameworks and libraries, including OpenCV, Tensorflow, and Darknet. The main application functionality spans eight Python modules.

4.1 Module tracking_app

This module is the main entry point of the application. It contains only one essential function that is responsible for running the proposed framework. In this function, we first load the input image stream, calibration parameters and other components such as object detectors, feature extractors, tracker, and image viewer. Then there is a loop that runs the algorithms as long as input frames are available. The application can be run either from an IDE or from the command line with various input parameters.

4.2 Module detection

The detection module contains multiple wrappers for various object and face detectors proposed in 3.3. It is responsible for loading these models based on input parameters, detecting objects in the input frame, and filtering detections by input criteria.

4.3 Module feature_extraction

In this module, we propose to interface and pre-processing to various appearance feature extractors including age, gender, and emotion models, but also height estimator. The module also contains a function that assigns face regions to people bounding boxes.

4.4 Module kalman_filter

This module is vital for the proposed motion analysis method because it contains the configuration of the utilized filter. Our approach uses Kalman Filter implementation from famous FilterPy [86] library. The module is also responsible for converting filter's internal state to visualizable bounding box and for converting measurements to filter's input.

4.5 Module matching

The matching module contains an efficient implementation of a proposed distance metric, namely the cosine metric for appearance (section 3.3.5.2) and enhanced Euclidean distance for the state (section 3.3.5.1). Metrics are calculated by NumPy, which is a high-performance linear algebra library written in C language. To make the computations as efficient as possible, we utilize the NumPy broadcasting technique.

4.6 Module tracking

In the tracking module, we propose PeopleTracker class that is responsible for the actual matching of detections and tracks. It is mostly inspired by code from [103], but some modifications in the matching algorithm were made. Successfully matched tracks are updated from here, and unmatched detections are transformed to tentative tracks.

4.7 Module track

An instance of Person class from the track module represents a single person that was detected in the scene. Each person instance has set of attributes including id, color for visualization, status, creation frame number, time since the last update, age in frame units, last body bounding box, last face bounding box, Kalman filter instance, and dictionary with gathered features.

4.8 Module output_statistics

This module is responsible for transforming all possible data to brief output statistics. It includes a class TrackingEvaluator which evaluates the tracking performance of the algorithm based on provided annotations. Class Output-Statistics is responsible for converting gathered data about people to usable outputs.

4.9 Module visualization

Visualization module contains all drawing related functions including plotting output graphs using Matplotlib and Seaborn libraries. It implements an ImageViewer class that is responsible for showing the preview of processed images. The ImageViewer class also contains functionality that allows easier debugging by allowing users to pause the algorithm or process the stream by one frame at a time.

Chapter 5

Evaluation

In this chapter, we firstly discuss the reason behind creating a new dataset. We further present achieved results with our proposed framework, and lastly, we propose additional improvements for future work.

5.1 Dataset

There are many public annotated datasets [2, 20, 22] for a person tracking task and many others targeting soft-biometrics [125, 137]. However, their focus is a little different than what is needed for the evaluation of this thesis. Firstly, individuals only appear in a few images of the sequence, which is not entirely the case for retail, where it is required to maintain the individual's long-term identity even in the case of multiple occlusions. Secondly, these datasets do not focus on the joint task of tracking and soft-biometrics extraction. The available tracking sequence lacks additional information about people and the scene parameters that are required to calibrate the camera. It was, therefore, advisable to create an in-the-wild dataset to simulate a real scenario.

The 14th floor of the FCE CTU building was used to create a dataset, where it was possible to place a camera to a place that is most similar to the retail case. In total, over 10,000 images were captured, of which 2463 frames were hand-annotated with person bounding-boxes and identities. There are a total of 7028 bounding boxes with 11 person identities. Face regions were not annotated because it would have no added value for the evaluation. Related individual's soft-biometrics data such as age, gender, and height are known, but ground-truth data for emotions and trajectories are not available. There are only male representatives in the dataset. Captured frames are not further pre-processed.

5.2 Hardware and software setup

The speed performance of algorithms is tightly dependent on the utilized hardware and software setup. The proposed methods were tested on the computer available in ImproLab. The testing setup consisted of Intel i5 - 7600K @ 3.80GHz, NVIDIA GeForce RTX 2080 TI, 16 GB RAM, and SSD disk. The software dependencies were CUDA 10, cuDNN 7.5, Tensorflow 1.13, OpenCV 3.4.2, and the latest version of Darknet. All third-party libraries were compiled for GPU support.

5.3 Tracking results

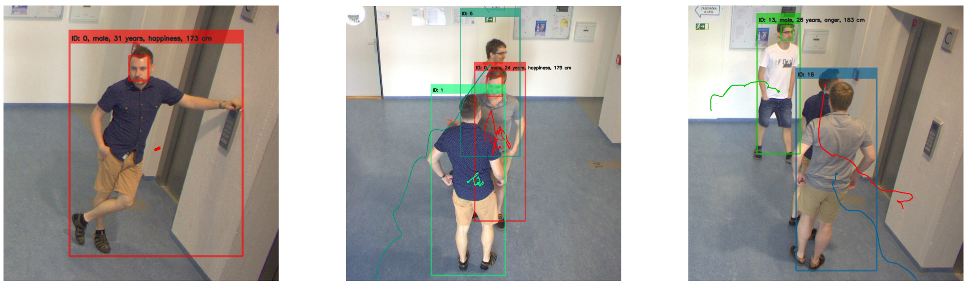
In order to be able to compare individual tracking solutions among each other, it is necessary to determine a suitable metric. Because metrics for object detection and tracking in the video can often be non-intuitive and complex, the well-known MOTP and MOTA metrics are adopted in this work. MOTP metrics focus on the quality of detected regions, while MOTA focuses on tracking accuracy, and its calculation affects the number of false positives (FP), false negatives (FN), and identity switches (IDSW). Factors affecting MOTA are also compared, as it is interesting to see how they change depending on the detection network. The metrics are described in more detail in section 1.5; however, to be concise, higher MOTA or MOTP means better.

We present our results in table 5.1 for different object detection architectures – YOLOv2, YOLOv3, YOLOv3 SPP, and Faster R-CNN with ResNet50 backbone. For the retail usage, the number of identity switches (IDSW) is a crucial metric, and in this aspect, the Faster R-CNN outperformed other detectors by a large margin. We also provide some output predictions in Fig. 5.1 and example video in [143] demonstrating robustness of the proposed algorithms.

	YOLOv2	**YOLOv3**	**YOLOv3 SPP**	**Faster R-CNN**
MOTP	0.795	0.84	0.89	0.87
MOTA	0.825	0.94	0.905	0.94
FN	879	306	470	228
FP	147	132	223	167
IDSW	14	8	10	2
FPS	26.62	22.2	16.29	17.37

Table 5.1: Tracker evaluation table.

In the table with results, we can see that the more complex the architecture employed, the more accurate the results, which again verified the fact presented in the introduction chapter, that a robust object detector is a key to proper tracking. Specifically, we can observe the number of missed detections


ID: 0, male, 31 years, happiness, 173 cm
ID: 6
ID: 0, male, 24 years, happiness, 175 cm
ID: 1
ID: 13, male, 26 years, anger, 183 cm
ID: 16

(a) (b) (c)

Figure 5.1: Image examples of our framework. (a) An example of a person facing the camera so his biometrics can be accurately estimated. Only the height value is inaccurate by 2 cm, which is a very precise outcome. (b) An example of people who are close to each other without affecting the tracker accuracy. (c) An example of people who are even closer than in (b). The person in the middle is fully occluded, and no detection is available for him. However, the red circle is trying to predict his position, in case he reappears in the scene. The prediction of soft-biometrics for the person with ID 13 is inaccurate by one year and 3 cm.

(FN) and identity switches (IDSW) is decreasing with more robust architecture. Faster R-CNN generally has more "ghost" detections (FP), but it was something we expected already in the Methodology chapter, and it is not such a problem that would significantly affect the results of the framework.

Interesting observations are worse results achieved by the modified YOLOv3 with spatial pyramid pooling (SPP) architecture. According to the original paper, it should work better on small objects, and although it provided more accurate bounding boxes (MOTP), all other aspects suggest that it is not very suitable for tracking algorithms.

The fact that FPS is decreasing with the complexity of object detector architecture is no surprise. Generally speaking, the bottle-neck of the tracker is the object detector. Based on our observations and measurements, the tracking algorithm without the object detector component can run at around 500 FPS. It is, therefore, necessary for any task to select the suitable object detector first. In our case, even with the complex Faster R-CNN, the tracking achieves near real-time performance.

5.4 Soft-biometrics results

The evaluation of soft-biometrics is a bit more complicated. It is not possible to obtain precise ground-truth for trajectories and emotions, so there is no room for evaluation. However, three more remaining features could be possibly evaluated – age, gender, and height. We decided to choose only age and height for the evaluation because we could not find a female representative at the time of the dataset creation.

The situation is even more complicated because some biometric information cannot always be extracted. It may happen that a person will always be back to the camera and it will not be possible to extract his facial information, or a person moves very fast or is occluded all the time, so it is not possible to estimate his height. Therefore, we only evaluate cases where it is possible to extract suitable soft-biometric data.

We use mean absolute error (MAE) metric described in section 1.5 for evaluation instead of root-mean-square error (RMSE) because it has easier interpretation. Description of the utilized detectors can be found in Methodology chapter.

In table 5.2, we present our outcome for age estimation for various face detectors. From the results, we can observe that the alignment feature in MTCNN has a great positive impact on accuracy, but it drastically reduces the FPS. Tensorflow Mobilenet SSD architecture provides the best trade-off between performance (FPS) and accuracy (Age MAE).

	faced	Mobilenet SSD	MTCNN
Age MAE	5.21	3.82	2.65
FPS	78.41	119.86	7.01

Table 5.2: Age estimation evaluation table.

The last table 5.3 focuses on the influence of object detector on height estimation. YOLO architectures generally provide less accurate bounding boxes, thus making the height error larger. The achieved results correspond to the MOTP outcome in table 5.1. We can also see that the speed performance degradation compared to the original FPS is negligible. To conclude, height estimation works well in most cases, and it has a little computational overhead.

	YOLOv2	YOLOv3	Faster R-CNN
Height MAE	7.31	5.84	4.09
FPS	26.35	21.97	17.17

Table 5.3: Height estimation evaluation table.

5.5 Further work

This complicated and broad task is not something that can be fully accomplished in one final thesis. Therefore, in this section, we suggest some improvements for future work.

5.5.1 Tracking improvements

Current state-of-the-art object detectors are already achieving outstanding results. People are detected in most cases when they are visible. The problem arises when people are entirely occluded, so they are not visible to the camera, but we still need to retain their identity when they show up. This is a situation where having robust tracker helps.

Our tracker can deal with short-term occlusions which are mostly situation when people pass by. However, it is not designed for the case when individuals are completely missing for a few seconds. To improve in this manner, the missing tracks would need to be kept in the memory for a more extended period than 50 frames, but also a new matching metric for these cases would need to be developed.

The tracking performance could also be improved by adding a non-linear state filter such as unscented Kalman filter or particle filter, which would improve the matching phase. The reason is that not all people behave linearly in their movement. Therefore, the current filter may occasionally diverge.

From our observation, utilized object detectors are already robust with default parameters and pre-trained models. Therefore, there is no reason for re-training them for a particular task with people. There could be significant speed-up improvement by proposing an object detector that can detect faces and people simultaneously.

5.5.2 Soft-biometrics extraction improvements

Our framework can extract face information (age, gender, emotion), height, and trajectory. This information can only be obtained under certain conditions. If no face is visible during the whole tracking session, then it is not possible to output any face information. If the object moves quickly, then extracted height information is not accurate. Moreover, if an object is occluded, then trajectory information might contain gaps.

Of course, all these shortcomings could be improved, but the significant improvement, for now, would be retraining the face extraction models on more in-the-wild data to improve the accuracy. There is no need to replace utilized architectures as they achieve state-of-the-art performance on current datasets, but it is needed to fine-tune the weights by faces that are captured at an angle, not only from the front view. It would also help to evaluate the methods on data from multiple environments and with more types of people.

Another improvement could be achieved by employing a model that can do face embedding. This embedding then could be used in the matching phase to recover long term occluded individuals.

Conclusion

This work was focused on building a framework for task of people tracking and soft-biometric data extraction. In the introductory chapter, we thoroughly described the assignment with sufficient motivation, but also with its challenges. In the following theoretical chapter, we have described the necessary theoretical background in detail for understanding of this thesis. In chapter 2, we briefly reviewed the existing solutions, and based on that some popular methods were implemented from scratch or customized from open-source repositories to meet the thesis goals. The implemented solution is then evaluated in the evaluation chapter, and further improvements are proposed. The result is a functioning people tracking and soft-biometrics extracting framework that can be deployed in real-world application.

According to the table 5.1, the proposed algorithms achieve state-of-the-art results in long-term people tracking. The best solution solution based on Faster R-CNN achieves only two identity switches on our dataset, which is a an outstanding outcome. The quality of the detected boxes is also very high. Evaluated height and age soft-biometrics achieve are very precise. However, in the upcoming work it is necessary to make an evaluation on more individuals.

The proposed methods are designed in several variations to meet different trade-offs of accuracy versus computational expensiveness, and since the final design of the framework is fully modular, it can be easily configured to meet demands for various other scenarios. We put much effort into the application design as a whole so that it can be easily expanded and it can now be deployed in real-world applications. There are still some shortcomings in the framework, but they are described so that they can be further worked on and we hope that this work will be a useful as a starting point for other people interested in this topic.

Bibliography

[1] Yao, A.; Gall, J.; et al. Interactive object detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 3242–3249.

[2] Leal-Taixé, L.; Milan, A.; et al. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv:1504.01942 [cs]*, Apr. 2015, arXiv: 1504.01942. Available from: `http://arxiv.org/abs/1504.01942`

[3] Ruffle, J. K.; Farmer, A. D.; et al. Artificial Intelligence-Assisted Gastroenterology—Promises and Pitfalls. *The American journal of gastroenterology*, 2018: p. 1.

[4] Kurama, V. Unsupervised Learning with Python. `https://towardsdatascience.com/unsupervised-learning-with-python-173c51dc7f03`, 2018, accessed 05/04/19.

[5] Wikipedia contributors. Reinforcement learning — Wikipedia, The Free Encyclopedia. `https://en.wikipedia.org/wiki/Reinforcement_learning`, 2019, [Accessed 05/04/19].

[6] Khandelwal, P. 7 Steps to Understanding Computer Vision. `https://www.kdnuggets.com/2016/08/seven-steps-understanding-computer-vision.html`, 2016, accessed 05/04/19.

[7] Levin, G. Image Processing and Computer Vision. `https://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html`, 2019, accessed 04/04/19.

[8] Furtado, H.; Campos Velho, H.; et al. Artificial Neural Networks Applied to the Lorenz Dynamical System via Variational Data Assimilation. 04 2019.

[9] Mathworks Collective. Convolutional Neural Network. https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html, 2019, accessed 04/04/19.

[10] Fei-Fei Li, S. Y., Justin Johnson. CS231n: Convolutional Neural Networks for Visual Recognition. http://cs231n.github.io, 2019, accessed 04/04/19.

[11] Xu, J. Deep Learning for Object Detection: A Comprehensive Review. https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9, 2017, accessed 05/04/19.

[12] Redmon, J.; Divvala, S.; et al. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[13] Wang, M.; Deng, W. Deep face recognition: A survey. *arXiv preprint arXiv:1804.06655*, 2018.

[14] Rosebrock, A. Intersection over Union for object detection. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection, 2016, accessed 04/04/19.

[15] Basler AG. Basler ace acA1920-50gc - Area Scan Camera. https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1920-50gc, accessed 04/04/19.

[16] Basler AG. Basler Lens C125-0818-5M F1.8 f8mm - Lenses. https://www.baslerweb.com/en/products/vision-components/lenses/basler-lens-c125-0818-5m-f1-8-f8mm, accessed 04/04/19.

[17] Rosebrock, A. Non-Maximum Suppression in Python. https://www.pyimagesearch.com/2015/02/16/faster-non-maximum-suppression-python, 2015, accessed 04/04/19.

[18] Hartley, R.; Zisserman, A. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[19] Criminisi, A. Single-view metrology: Algorithms and applications. In *Joint Pattern Recognition Symposium*, Springer, 2002, pp. 224–239.

[20] Ferryman, J.; Shahrokni, A. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, IEEE, 2009, pp. 1–6.

[21] Milan, A.; Leal-Taixé, L.; et al. MOT16: A Benchmark for Multi-Object Tracking. *arXiv:1603.00831 [cs]*, Mar. 2016, arXiv: 1603.00831. Available from: http://arxiv.org/abs/1603.00831

[22] Zhu, P.; Wen, L.; et al. Vision Meets Drones: A Challenge. *arXiv preprint arXiv:1804.07437*, 2018.

[23] Wikipedia contributors. Biometrics — Wikipedia, The Free Encyclopedia. `https://en.wikipedia.org/w/index.php?title=Biometrics`, 2019, accessed 04/04/19.

[24] Ristani, E.; Solera, F.; et al. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. In *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

[25] Luo, W.; Xing, J.; et al. Multiple object tracking: A literature review. *arXiv preprint arXiv:1409.7618*, 2014.

[26] Fan, L.; Wang, Z.; et al. A survey on multiple object tracking algorithm. In *2016 IEEE International Conference on Information and Automation (ICIA)*, IEEE, 2016, pp. 1855–1862.

[27] Emami, P.; Pardalos, P. M.; et al. Machine learning methods for solving assignment problems in multi-target tracking. *arXiv preprint arXiv:1802.06897*, 2018.

[28] Konushin, A. Deep Learning in Computer Vision. `https://www.coursera.org/lecture/deep-learning-in-computer-vision/examples-of-multiple-object-tracking-methods-VJZUW`, 2017, accessed 04/04/19.

[29] Bewley, A.; Ge, Z.; et al. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016, pp. 3464–3468.

[30] Bergmann, P.; Meinhardt, T.; et al. Tracking without bells and whistles. *CoRR*, volume abs/1903.05625, 2019, `1903.05625`. Available from: `http://arxiv.org/abs/1903.05625`

[31] Dobrev, D. A Definition of Artificial Intelligence. *arXiv preprint arXiv:1210.1568*, 2012.

[32] Hutter, M. *Universal artificial intelligence: Sequential decisions based on algorithmic probability.* Springer Science & Business Media, 2004.

[33] Faggella, D. What is Machine Learning? `https://emerj.com/ai-glossary-terms/what-is-machine-learning`, 2012, accessed 05/04/19.

[34] Pearson, K. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, volume 2, no. 11, 1901: pp. 559–572.

[35] Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

[36] Shalev-Shwartz, S.; Ben-David, S. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[37] Witten, I. H.; Frank, E.; et al. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[38] Cortes, C.; Vapnik, V. Support-vector networks. *Machine learning*, volume 20, no. 3, 1995: pp. 273–297.

[39] Chapelle, O.; Scholkopf, B.; et al. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, volume 20, no. 3, 2009: pp. 542–542.

[40] Johnson, S. C. Hierarchical clustering schemes. *Psychometrika*, volume 32, no. 3, 1967: pp. 241–254.

[41] MacQueen, J.; et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, Oakland, CA, USA, 1967, pp. 281–297.

[42] Ester, M.; Kriegel, H.-P.; et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, 1996, pp. 226–231.

[43] Sutton, R. S.; Barto, A. G.; et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[44] François-Lavet, V.; Henderson, P.; et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, volume 11, no. 3-4, 2018: pp. 219–354.

[45] Goodfellow, I.; Bengio, Y.; et al. *Deep learning*. MIT press, 2016.

[46] LeCun, Y.; Bengio, Y.; et al. Deep learning. *nature*, volume 521, no. 7553, 2015: p. 436.

[47] Lee, H.; Grosse, R.; et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009, pp. 609–616.

[48] Mikolov, T.; Karafiát, M.; et al. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[49] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, volume 36, no. 4, 1980: pp. 193–202.

[50] LeCun, Y.; Boser, B.; et al. Backpropagation applied to handwritten zip code recognition. *Neural computation*, volume 1, no. 4, 1989: pp. 541–551.

[51] LeCun, Y.; Bottou, L.; et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, volume 86, no. 11, 1998: pp. 2278–2324.

[52] Russakovsky, O.; Deng, J.; et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, volume 115, no. 3, 2015: pp. 211–252.

[53] Koch, C. How the Computer Beat the Go Master. `https://www.scientificamerican.com/article/how-the-computer-beat-the-go-master`, 2016, accessed 05/04/19.

[54] Zheng, L. A survey and critique of deep learning on recommender systems. *Chicago: University of Illinois*, 2016.

[55] Szeliski, R. *Computer Vision: Algorithms and Applications*. Berlin, Heidelberg: Springer-Verlag, first edition, 2010, ISBN 1848829345, 9781848829343.

[56] Koen, S. Architecting a Machine Learning Pipeline. `https://towardsdatascience.com/architecting-a-machine-learning-pipeline-a847f094d1c7`, 2019, accessed 05/04/19.

[57] Viola, P.; Jones, M.; et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, volume 1, 2001: pp. 511–518.

[58] Ojala, T.; Pietikäinen, M.; et al. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, , no. 7, 2002: pp. 971–987.

[59] Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, IEEE Computer Society, 2005, pp. 886–893.

[60] Lowe, D. G. Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image. Mar. 23 2004, uS Patent 6,711,293.

[61] Bay, H.; Tuytelaars, T.; et al. Surf: Speeded up robust features. In *European conference on computer vision*, Springer, 2006, pp. 404–417.

[62] Zurada, J. M. *Introduction to artificial neural systems*, volume 8. West publishing company St. Paul, 1992.

[63] Rosenblatt, F. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[64] Hornik, K.; Stinchcombe, M.; et al. Multilayer feedforward networks are universal approximators. *Neural networks*, volume 2, no. 5, 1989: pp. 359–366.

[65] Hecht-Nielsen, R. Theory of the backpropagation neural network. In *Neural networks for perception*, Elsevier, 1992, pp. 65–93.

[66] Krizhevsky, A.; Sutskever, I.; et al. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[67] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[68] Szegedy, C.; Vanhoucke, V.; et al. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[69] He, K.; Zhang, X.; et al. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[70] Xie, S.; Girshick, R.; et al. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[71] Huang, G.; Liu, Z.; et al. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[72] Das, S. CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more. `https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5`, 2017, accessed 04/04/19.

[73] Jordan, J. Common architectures in convolutional neural networks. `https://www.jeremyjordan.me/convnet-architectures/`, 2018, accessed 04/04/19.

[74] Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[75] He, K.; Zhang, X.; et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[76] Kalman, R. E. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, volume 82, no. 1, 1960: pp. 35–45.

[77] Horn, B. K.; Schunck, B. G. Determining optical flow. *Artificial intelligence*, volume 17, no. 1-3, 1981: pp. 185–203.

[78] Piccardi, M. Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 4, IEEE, 2004, pp. 3099–3104.

[79] Levinson, J.; Askeland, J.; et al. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2011, pp. 163–168.

[80] Zhang, Z.; et al. Flexible camera calibration by viewing a plane from unknown orientations. In *Iccv*, volume 99, 1999, pp. 666–673.

[81] Criminisi, A.; Reid, I.; et al. Single view metrology. *International Journal of Computer Vision*, volume 40, no. 2, 2000: pp. 123–148.

[82] Orteu, J.-J.; Garric, V.; et al. Camera calibration for 3D reconstruction: application to the measurement of 3D deformations on sheet metal parts. In *New Image Processing Techniques and Applications: Algorithms, Methods, and Components II*, volume 3101, International Society for Optics and Photonics, 1997, pp. 252–264.

[83] Wilczkowiak, M.; Boyer, E.; et al. Camera calibration and 3D reconstruction from single images using parallelepipeds. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, IEEE, 2001, pp. 142–148.

[84] Kushal, A.; Bansal, V.; et al. A Simple Method for Interactive 3D Reconstruction, Camera Calibration from a Single View. In *ICVGIP*, Citeseer, 2002.

[85] Criminisi, A. *Accurate visual metrology from single and multiple uncalibrated images*. Springer Science & Business Media, 2012.

[86] Labbe, R. Kalman and bayesian filters in python. 2015.

[87] Ouaknine, A. Review of Deep Learning Algorithms for Object Detection. `https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852`, 2017, accessed 05/04/19.

[88] Huang, J.; Rathod, V.; et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[89] Girshick, R.; Donahue, J.; et al. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, volume 38, no. 1, 2016: pp. 142–158.

[90] Uijlings, J. R.; Van De Sande, K. E.; et al. Selective search for object recognition. *International journal of computer vision*, volume 104, no. 2, 2013: pp. 154–171.

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

[92] Ren, S.; He, K.; et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 2015, pp. 91–99.

[93] Taigman, Y.; Yang, M.; et al. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[94] Hayes, B. Most Used Data Science Tools and Technologies in 2017 and What to Expect for 2018. `http://businessoverbroadway.com/2018/01/02/most-used-data-science-tools-and-technologies-in-2017-and-what-to-expect-for-2018`, 2018, accessed 04/04/19.

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

[96] Redmon, J. Darknet: Open Source Neural Networks in C. `http://pjreddie.com/darknet/`, 2013–2016.

[97] Abadi, M.; Barham, P.; et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[98] Paszke, A.; Gross, S.; et al. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.

[99] Caffe2 Development Team. Caffe2 - lightweight, modular, and scalable deep learning framework. `https://caffe2.ai`, 2019, accessed 04/04/19.

[100] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, volume abs/1605.02688, May 2016. Available from: `http://arxiv.org/abs/1605.02688`

[101] Hamers, L.; et al. Similarity measures in scientometric research: The Jaccard index versus Salton's cosine formula. *Information Processing and Management*, volume 25, no. 3, 1989: pp. 315–18.

[102] Willmott, C. J.; Matsuura, K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research*, volume 30, no. 1, 2005: pp. 79–82.

[103] Wojke, N.; Bewley, A.; et al. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2017, pp. 3645–3649.

[104] Milan, A.; Rezatofighi, S. H.; et al. Online multi-target tracking using recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[105] Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural computation*, volume 9, no. 8, 1997: pp. 1735–1780.

[106] Sadeghian, A.; Alahi, A.; et al. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 300–311.

[107] Voigtlaender, P.; Krause, M.; et al. MOTS: Multi-Object Tracking and Segmentation. *CoRR*, volume abs/1902.03604, 2019, `1902.03604`. Available from: `http://arxiv.org/abs/1902.03604`

[108] Fookes, C.; Denman, S.; et al. Semi-supervised intelligent surveillance system for secure environments. In *2010 IEEE International Symposium on Industrial Electronics*, IEEE, 2010, pp. 2815–2820.

[109] Quintana, M.; Menéndez, J. M.; et al. Improving retail efficiency through sensing technologies: A survey. *Pattern Recognition Letters*, volume 81, 2016: pp. 3–10.

[110] De Carolis, B.; Macchiarulo, N.; et al. A Comparative Study on Soft Biometric Approaches to Be Used in Retail Stores. In *International Symposium on Methodologies for Intelligent Systems*, Springer, 2018, pp. 120–129.

[111] Viswanath, P.; Kakadiaris, I.; et al. A simplified error model for height estimation using a single camera. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, IEEE, 2009, pp. 1259–1266.

[112] Momeni-k, M.; Diamantas, S. C.; et al. Height Estimation from a Single Camera View. In *VISAPP (1)*, 2012, pp. 358–364.

[113] Li, S.; Nguyen, V. H.; et al. A simplified nonlinear regression method for human height estimation in video surveillance. *EURASIP Journal on Image and Video Processing*, volume 2015, no. 1, 2015: p. 32.

[114] Yamaguchi, K.; Kiapour, M. H.; et al. Parsing clothing in fashion photographs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 3570–3577.

[115] Liu, Z.; Luo, P.; et al. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1096–1104.

[116] He, K.; Gkioxari, G.; et al. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[117] Ge, Y.; Zhang, R.; et al. DeepFashion2: A Versatile Benchmark for Detection, Pose Estimation, Segmentation and Re-Identification of Clothing Images. *arXiv preprint arXiv:1901.07973*, 2019.

[118] Balaban, S. Deep learning and face recognition: The state of the art. In *Biometric and Surveillance Technology for Human and Activity Identification XII*, volume 9457, International Society for Optics and Photonics, 2015, p. 94570B.

[119] Yin, X.; Liu, X. Multi-task convolutional neural network for pose-invariant face recognition. *IEEE Transactions on Image Processing*, volume 27, no. 2, 2018: pp. 964–975.

[120] Ranjan, R.; Patel, V. M.; et al. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 41, no. 1, 2019: pp. 121–135.

[121] Ranjan, R.; Sankaranarayanan, S.; et al. An all-in-one convolutional neural network for face analysis. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, IEEE, 2017, pp. 17–24.

[122] Wang, X.; Guo, R.; et al. Deeply-learned feature for age estimation. In *2015 IEEE Winter Conference on Applications of Computer Vision*, IEEE, 2015, pp. 534–541.

[123] Rothe, R.; Timofte, R.; et al. Dex: Deep expectation of apparent age from a single image. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 10–15.

[124] Yang, T.-Y.; Huang, Y.-H.; et al. SSR-Net: A Compact Soft Stagewise Regression Network for Age Estimation. In *IJCAI*, 2018, pp. 1078–1084.

[125] Goodfellow, I. J.; Erhan, D.; et al. Challenges in representation learning: A report on three machine learning contests. In *International Conference on Neural Information Processing*, Springer, 2013, pp. 117–124.

[126] Arriaga, O.; Valdenegro-Toro, M.; et al. Real-time convolutional neural networks for emotion and gender classification. *arXiv preprint arXiv:1710.07557*, 2017.

[127] ThoughtWorks Arts development team. EmoPy - A deep neural net toolkit for emotion analysis via Facial Expression Recognition. `https://github.com/thoughtworksarts/EmoPy`, 2018, accessed 04/04/19.

[128] Zeng, N.; Zhang, H.; et al. Facial expression recognition via learning deep sparse autoencoders. *Neurocomputing*, volume 273, 2018: pp. 643–649.

[129] Sun, W.; Shao, S.; et al. A sparse auto-encoder-based deep neural network approach for induction motor faults classification. *Measurement*, volume 89, 2016: pp. 171–178.

[130] Ruiz, N.; Chong, E.; et al. Fine-Grained Head Pose Estimation Without Keypoints. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[131] Tsun-Yi Yang, Y.-Y. L. Y.-Y. C., Yi-Ting Chen. FSA-Net: Learning Fine-Grained Structure Aggregation for Head Pose Estimation from a Single Image. 2019.

[132] Basler AG. Basler pylon. `https://github.com/basler/pypylon`, accessed 04/04/19.

[133] Jonker, R.; Volgenant, A. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, volume 38, no. 4, 1987: pp. 325–340.

[134] Lin, T.-Y.; Maire, M.; et al. Microsoft coco: Common objects in context. In *European conference on computer vision*, Springer, 2014, pp. 740–755.

[135] Choi, E.; Lee, C. Feature extraction based on the Bhattacharyya distance. *Pattern Recognition*, volume 36, no. 8, 2003: pp. 1703–1709.

[136] Itzcovich, I. Near Real Time CPU Face detection using deep learning. `https://github.com/iitzco/faced`, 2018, accessed 04/04/19.

[137] Yang, S.; Luo, P.; et al. WIDER FACE: A Face Detection Benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[138] Itzcovich, I. Tensorflow Face Detector. `https://github.com/yeephycho/tensorflow-face-detection`, 2017, accessed 04/04/19.

[139] Liu, W.; Anguelov, D.; et al. Ssd: Single shot multibox detector. In *European conference on computer vision*, Springer, 2016, pp. 21–37.

[140] Zhang, K.; Zhang, Z.; et al. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, volume 23, no. 10, Oct 2016: pp. 1499–1503, ISSN 1070-9908, doi:10.1109/LSP.2016.2603342.

[141] de Paz Centeno, I. MTCNN face detection implementation for TensorFlow. `https://github.com/ipazc/mtcnn`, 2018, accessed 04/04/19.

[142] Uchida, Y. Keras implementation of a CNN network for age and gender estimation. `https://github.com/yu4u/age-gender-estimationn`, 2017, accessed 04/04/19.

[143] ImproLab Team. Tracking video example. `https://www.youtube.com/watch?v=D-Z1VHqBqcE`, 2019, accessed 04/04/19.

Appendix A

Acronyms

AI artificial intelligence

ANN artificial neural network

CMOS complementary metal oxide semiconductor

CNN convolutional neural network

CTU Czech Technical University

CV computer vision

DEX Deep EXpectation

DL deep learning

DLT Direct Linear Transform

DSAE deep sparse autoencoders

Fast R-CNN fast region-based convolutional network

Faster R-CNN faster region-based convolutional network

FCE Faculty of Civil Engineering

FIT Faculty of Information Technology

FN false negatives

FP false positives

FPS frames per second

GigE Gigabit Ethernet

GPU	graphics processing unit
HOG	histogram of oriented gradients
HSV	hue-saturation-value
IDSW	identity switches
ImproLab	Image Processing Laboratory
IoU	intersection over union
LBP	local binary pattern
LSTM	long short-term memory
MAE	mean absolute error
ML	machine learning
MLP	multi layer perceptron
MOT	multiple object tracking
MOTA	multiple object tracking accuracy
MOTP	multiple object tracking precision
MOTS	multiple object tracking and segmentation
MP	megapixel
MTMCT	multi-target multi-camera tracking
NMS	non-maximum suppression
NN	neural network
PCA	principal component analysis
R-CNN	region-based convolutional network
ReID	re-identification
RGB	red-green-blue
RMSE	root-mean-square error
RNN	recurrent neural network

RoI	region of interest
RPN	region proposal network
SDK	software development kit
SIFT	Scale-invariant feature transform
SORT	simple online and real-time tracking
SPP	spatial pyramid pooling
SSD	single-shot detector
SSR-Net	Soft Stagewise Regression Network
SURF	Speeded-Up Robust Features
SVM	support vector machine
YOLO	you only look once

Appendix **B**

Media contents

```
│ readme.txt ...................... the file with CD contents description
├── data ........................................ the data files directory
│   ├── example_sequence . the directory with example sequence from dataset
│   │   └── *.jpg .................................. the example images
│   ├── example_sequence_graphs .....the directory of graphs of experiments
│   │   └── *.png ............................. the motion output graphs
│   └── tracking_example.mp4 ..................... the example video file
├── src ................................... the directory of source codes
│   ├── models ...................... the directory of deep learning modules
│   ├── utils ............................ the directory of helper modules
│   └── *.py ..................................... the Python source files
└── text ...................................... the thesis text directory
    ├── thesis ............ the directory of LaTeX source codes of the thesis
    └── thesis.pdf ..................... the Diploma thesis in PDF format
```