Master Thesis

Shopfloor Monitor: Multi-Camera-Based Detection and Tracking System for a Manufacturing Environment

Lukas Bommes
Matriculation Number: 4367361

Technische Universität Braunschweig
Faculty of Mechanical Engineering
Institute of Machine Tools and Production Technology

and

Agency for Science, Technology and Research
Singapore Institute of Manufacturing Technology
Department of Manufacturing Execution and Control

Examiner: Prof. Dr.-Ing. Christoph Herrmann
Supervisors: Dr. Lee Kee Jin, Artem Turetskyy M.Sc.

Submitted on 10th March 2019

Declaration of Authorship

Abstract

Objective: This thesis aims at implementing and evaluating the *Shopfloor Monitor*, a software for multi-camera-based tracking of humans in a manufacturing environment and real-time visualization of the results in a virtual 2D floorplan.

Implementation Methods: Humans are tracked in multiple parallel video streams by means of multiple single-target trackers, such as MOSSE, KCF or CSRT. A deep learning-based object detector is used to localize humans in regularly sampled video frames. The detection results are then matched with existing trackers to update the tracker state and prevent drift of the tracker. This matching is performed by maximizing the total intersection over union (IoU) of detected bounding boxes and boxes predicted by the existing trackers. Here, the Hungarian algorithm is used. A low-pass filter is used to reduce the effect of false positive and false negative detections on the tracking results and stabilize the assigned target IDs. Camera calibration provides intrinsic camera parameters and homography matrices between the image planes and the factory ground-plane. These are used to transform the target locations from image coordinates to the factory floor. The target locations are stored in a database and sent to a web app for real-time visualization on a 2D floorplan of the factory.

Evaluation Methods: To evaluate accuracy and processing speed of the tracking system, a systematic parameter study is designed. A script automatically varies 7 hyperparameters, including the object detection and tracking algorithms, and reports the results. Tracking performance is quantified by the processing frame rate and by the metrics specified in the MOT benchmark. These metrics are computed on a specifically created video dataset, which contains 5 video sequences with a total of 888 frames and 12192 manually annotated targets.

Findings: The study provided deep insights into the developed detection and tracking system, which helped to choose three different sets of hyperparameters, yielding either a more accurate or a faster tracking system. The study also showed that the developed Shopfloor Monitor is fully functional and provides reasonable performance. State-of-the-art tracking performance was not reached. However, the work presented new application possibilities for such a system in manufacturing technology and successfully implemented a demonstrator for the selected use case of an immersive real-time visualization.

Keywords: Manufacturing, Object Detection, Object Tracking, Multi-Camera Multi-Target Tracking, Bounding Box, Camera Calibration, Visualization, Augmented Reality, Video Surveillance, Deep Learning, Machine Learning, Big Data, Database, Websocket, Web Programming, Software Architecture, Multi-Processing, Multi-Threading, GPU Computing, Python, Javascript, OpenCV, Tensorflor, MySQL, Hungarian Algorithm, Parameter Study, Dataset, Labeling, SORT, DeepSORT, MOSSE, KCF, CSRT, Faster R-CNN, R-FCN, SSD

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Modern smart factories are highly sophisticated cyberphysical systems, which generate large amounts of process- and operations-related data. This data is utilized by powerful data analytics tools, for example machine learning, to analyze, control and optimize manufacturing processes, and to drive business decisions [2–4]. Although the smart factory concept strives to fully automate the entire manufacturing process, human workers still play a major role in today's factories because of their superior abilities in terms of creativity, flexibility, problem-solving and learning of new tasks, which are unmatched by machines. However, there are also downsides to employing human workers. For example lower reliability, reproducibility, limited attention span and easy distractability, which can eventually lead to failures that cause expensive downtimes and loss in product quality [5]. To reduce human error and ensure a smooth operation of the manufacturing plant, it is of paramount importance to consider humans as an important factor of the manufacturing process. Unfortunately, humans are seldomly considered in modern factory concepts. Usually, only plant-, machine- and operations-related data is collected and made available to subsequent data analytics and control procedures. Monitoring human activity enables a company to keep up with competitors in terms of productivity, cost and time-to-market of new products because it helps to improve manufacturing and business processes. This eventually determines the success or failure of the enterprise [6].

The weak utilization of human-related data in modern manufacturing environments is lost potential and thus motivates the initiation of a novel research project. It aims at collecting human-related data, and places humans as an additional factor into the loop of process analytics, control and simulation. The research question of this project is as follows.

Research Question: How can a system (called the Shopfloor Monitor) be implemented which acquires and processes human-related data in manufacturing environments in real-time using only a network of multiple surveillance cameras?

Restricting the sensory input to multiple surveillance cameras, ensures minimum cost, time and effort when deploying the system at the customer's facility, and thus eases customer acceptance for the Shopfloor Monitor. Especially, if a network of surveillance cameras is already existent in a factory, no further hardware or installation cost arises. Furthermore, a wide range of state-of-the-art computer vision and data analytics tools are available and enable accurate localization of human workers on the shopfloor. As compared to other sensory systems cameras enable, for example, identification of workers or activity recognition.

To explore and answer the research question the following goals are defined for this thesis:

- Develop a specification for the Shopfloor Monitor.

- Review literature related to the specified system.
- Explain fundamentals of visual object detection and tracking needed for the implementation.
- Develop and implement a full-fledged prototype of the Shopfloor Monitor.
- Conduct experiments to evaluate performance of the Shopfloor Monitor and to understand the impact of various sub-components on accuracy and processing speed.
- Present and critically discuss the experimental results.
- Explore possible applications of the system in the manufacturing domain.

To better understand the goal of the Shopfloor Monitor the specification developed in chapter 2 has to be preempted. In the specification the scope of the Shopfloor Monitor is reduced to the task of tracking humans in multiple parallel video streams of a manufacturing facility, visualizing their locations in real-time on a 2D floorplan of the factory and storing the acquired trajectories for later analysis in a database. To achieve this goal, the Shopfloor Monitor uses state-of-the-art deep learning algorithms for the detection of humans in subsequent video frames, and modern tracking algorithms to associate detections across time and maintain the identity of tracked persons. In doing so, the developed Shopfloor Monitor pushes the state-of-the-art of solutions for multi-camera multi-target tracking and visualization, which are analyzed in chapter 4, and which use outdated detection and tracking algorithms. Apart from implementing the detection and tracking system, goal of this thesis is also to develop the overall architecture of the Shopfloor Monitor, in a way that allows easy extensibility of the system with further features, such as activity and face recognition. The visualization app is developed as part of a web-based app-framework, that allows for device- and location-independent access of the acquired data and easy expandability with future applications. The experimental evaluation of the Shopfloor Monitor makes use of the publicly available and commonly used Multiple Object Tracking (MOT) benchmark [7] in combination with an application-specific dataset. Methodically, an automated parameter study is conducted, which systematically varies predefined hyperparameters of the detection and tracking system and examines the resulting tracking accuracy and processing frame rate.

This work merges multiple sub-fields of computer science and manufacturing technology. While the application relates to smart manufacturing and cyber-physical systems, the implementation utilizes methods of camera calibration, visual object detection and tracking, and more specifically multi-camera multi-target tracking. System implementation is mainly an engineering task and requires knowledge about software architecture, the Tensorflow deep learning framework, the OpenCV library for video processing, Python based web servers, web development in Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and Javascript and the Web Graphics Library (WebGL) framework for web-based visualization. This thesis is addressed to researchers, students and software engineers who want to implement a multi-camera mulit-target tracking system which uses state-of-the-art methods for detection and tracking. In this case chapter 5 might be most interesting. People who are mainly interested in the applications of such a system without caring about the implementation, might skip chapter 5 and chapter 6 and focus on chapter 8.

The thesis is organized as follows. In chapter 2 requirements for the Shopfloor Monitor are derived and a specification of the system is developed. Chapter 3 explains the fundamentals of visual object detection and tracking that are needed to understand the subsequent chapters. In chapter 4 a literature review of publications closely related to the Shopfloor Monitor is performed. Chapter 5 describes the implementation of the Shopfloor Monitor in a top-down manner by breaking down the high-level system architecture into smaller sub-components. It provides details about the detection and tracking framework, the visualization web app and the camera calibration. In chapter 6 experiments on the detection and tracking module of the Shopfloor Monitor are conducted and results are presented. These results are thoroughly discussed in chapter 7. Possible applications of the Shopfloor Monitor in the context of manufacturing are introduced in chapter 8. Afterwards, privacy concerns arising from the use of the Shopfloor Monitor are presented in chapter 9. Modifications of the developed system are described to enable compliance with strict data privacy regulations of some countries. Chapter 10 concludes the research conducted in this work. It provides a critical review of the achievements and proposes future research directions.

Chapter 2

Use Case

This chapter aims at deriving a specification for the Shopfloor Monitor which is developed in the course of this work. First, the deployment environment of the system is analyzed in section 2.1. Here, special focus is laid on the practical challenges for development and operation of the system. Based on the identified challenges and the initial research question, requirements are discussed in section 2.2 and a specification of the Shopfloor Monitor is derived in section 2.3.

2.1 Environment Analysis

Before a specification of the Shopfloor Monitor can be developed it is necessary to analyze the environment in which the system is set up and identify associated challenges.

Even though the Shopfloor Monitor can be deployed to arbitrary sites, its intended use is in manufacturing. Thus, some assumptions about the environment can be made. For example, most companies already have a network of surveillance cameras installed or installation of a camera network is possible. These camera networks usually provide coverage of large areas of the manufacturing site and camera streams are aggregated by a central device, such as a network video recorder (NVR) or a PC. In many cases, IP cameras are used which can be directly accessed by their IP address. Furthermore, manufacturing sites are usually well illuminated and the brightness is constant. This simplifies data acquisition by means of cameras.

The described environment poses several challenges for the development of a multi-camera-based multi-target tracking system. Since an existing camera network shall be used, if available, no assumption about the exact camera model, the optical characteristics, frame rate, resolution, compression algorithm or transfer protocol can be made. Often network cameras are used which can be subject to connection problems and low network bandwidth, limiting image quality and introducing severe jitter. Thus, the Shopfloor Monitor must not rely on guaranteed arrival times of camera frames and be able to handle delayed or dropped frames and disconnected cameras. The cameras might also not remain static throughout system operation. In many cases pan-tilt-zoom (PTZ) cameras are used which can be moved via remote commands. This is challenging, because a mapping between image coordinates of localized targets and their true position in world coordinates can not easily be provided or has to be updated in real-time. Most surveillance cameras also do not provide any calibration parameters which are needed to compute such a mapping. Thus, calibration parameters must be determined by an extensive calibration procedure. Other challenges of using existing surveillance cameras are a potentially incomplete coverage of the manufacturing site, sub-optimal placement of cameras and occlusion of areas by obstacles. In other cases the view fields of multiple cameras may overlap. Problematic is also that the target's resolution depends on its distance to the camera. Targets further away from the camera have a lower resolution and, thus, it is more difficult to accurately infer knowledge about the target. Further problems arise from the fact that motion and activity of human targets are unpredictable and diverse. The appearance of individuals can change quickly and the number of people within a camera's field of view (FOV) can vary greatly. In crowded scenarios people are likely to be occluded by objects or other people. There might also be additional distractors, such as autonomous vehicles or other moving objects. Challenging is also that acquisition and processing of human-related data have to take place in real-time. This poses a minimal constraint on the computing hardware which competes with the goal of providing a low-cost solution.

2.2 System Requirements

Having analyzed the deployment environment and accompanying challenges, requirements for the Shopfloor Monitor can now be derived. There are several competing requirements:

- High *accuracy* of the acquired data.
- High *speed* of data acquisition and processing.
- Low overall *cost*.
- *Modularity* and *scalability* of the system architecture.

Achieving high accuracy relates to finding solutions for the challenges of incomplete site coverage, overlapping camera FOVs, occlusion, low target resolution, appearance changes of targets, crowded scenes and visual distractors. Solving these issues is mainly a question of the algorithms used to extract knowledge from the raw camera footage.

High acquisition and processing speed is needed to maximize throughput and minimize latency of the data extraction and to prevent data loss. High speed also allows to accurately detect events of short duration or high frequency, such as fast moving objects. This is especially crucial if the extracted data is used to drive safety-relevant systems, for example in fence-less robot operation or control of autonomous vehicles.

Achieving low cost eases customer acceptance and is an important factor for selling the final product. To keep the cost low, the Shopfloor Monitor has to utilize existing infrastructure, such as already installed surveillance cameras, and must not require installation of additional hardware. Furthermore, reliability of the system has to be high to reduce maintenance cost and cost due to breakdown or malfunction.

Modularity ensures easy extension of the system with further functions while maximizing utilization of existing software components. It also means, the system can be individualized based on the needs and the application of a customer and it widens the range of possible applications. Scalability implies a system architecture which allows to observe arbitrarily large manufacturing sites and arbitrary size of the camera network. Additionally, the system must be able to cope with a varying number people in each camera FOV.

Further requirements for the Shopfloor Monitor are compatibility with industrial safety standards and high software security. Moreover, a standard Linux operating system should be used to implement the system. This makes many reliable and mature software frameworks and tools available and simplifies the development process as well as system maintenance.

With regard to user interaction the Shoopfloor Monitor has to provide an easy-to-use interface for configuration of the system and visualization of the extracted data. This interface must be platform-independent and also run on mobile devices. Extracted data has to be stored long-term and must be easily accessible by other applications.

2.3 System Specification

Based on the list of requirements the specification of the Shopfloor Monitor can be developed. It is shown in fig. 2.1.

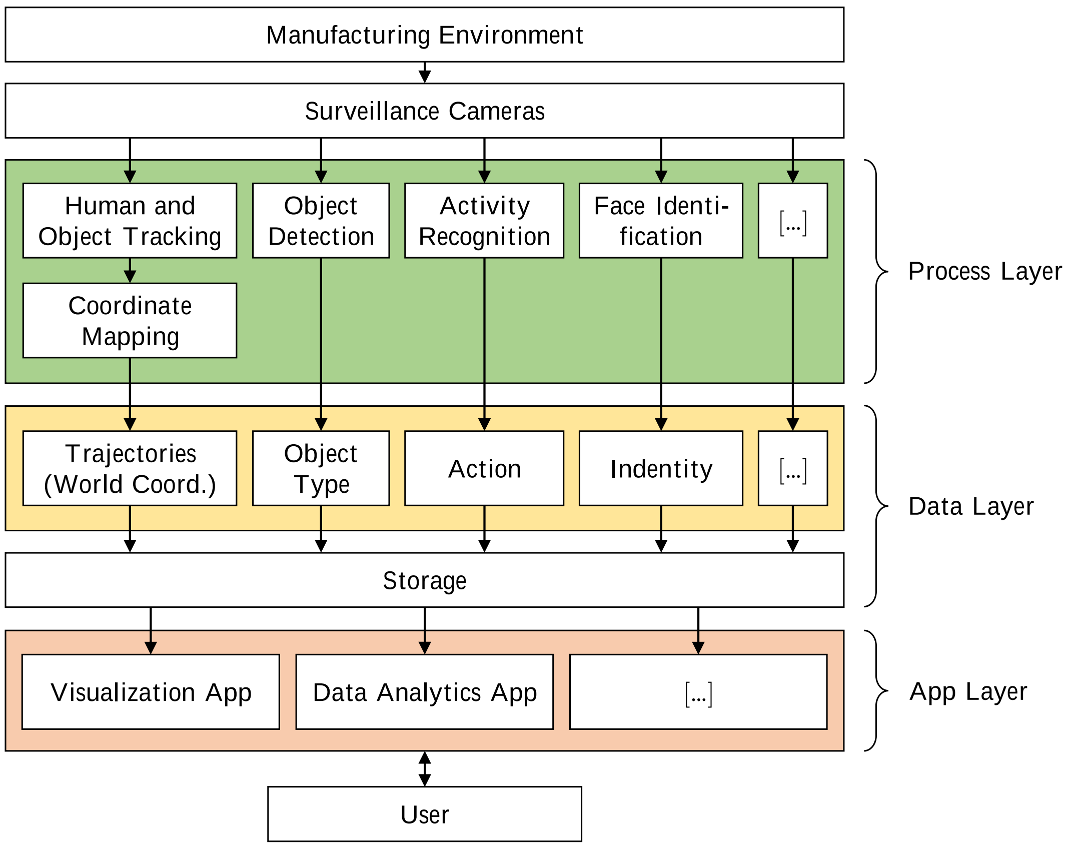


Fig. 2.1: Specification of the Shopfloor Monitor. Using a network of surveillance cameras, diverse human-related data can be extracted from the camera streams by applying state of the art methods of visual object detection and tracking, human activity recognition and face identification, enabling a broad range of applications.

The system has a layered architecture with the manufacturing environment and the surveillance cameras at the top level. The process layer combines several state-of-the-art algorithms to extract human-related data from the incoming raw video streams. These algorithms have to satisfy the requirements of high speed, high accuracy and scalability. Extracted data is aggregated and stored in the data layer and passed on to the app layer. The app layer contains web apps which utilize the extracted data for various tasks, such as indoor positioning, control of safety systems, accident detection or vehicle and robot control.

The layered design of the system and partitioning of tasks into modules ensures the required modularity and scalability. Customers can decide for which application they need the Shopfloor Monitor and, correspondingly, only modules needed for this application are activated. Scalability can be achieved by copying modules or entire layers onto multiple processing PCs.

Since development of such a broad spectrum of algorithms and apps exceeds the scope of this work, only one application is chosen to be realized in the next chapters. This application is the detection and tracking of humans and visualization of their trajectories on a virtual floorplan of the factory. One reason for choosing this application is the need for high processing speeds with rates close to the camera frame rate and the demand for high accuracy of the extracted positions. If this application can be realized, so can other with lower speed and accuracy requirements. Furthermore, a virtual floorplan is easily presentable and highlights the capabilities and opportunities of the Shopfloor Monitor in an illustrative way. Finally, a lot of algorithms for detection and tracking are available which simplifies and speeds up the development process.

Apart from restricting the scope to a single application, further simplifying assumptions are made. Overlapping camera FOVs are neglected and static cameras are assumed. Furthermore, only two cameras are utilized to reduce the computational demand and allow for processing on a single PC.

2.4 Conclusion

In this chapter a specification of the Shopfloor Monitor was developed. First, the deployment environment was analyzed and challenges posed onto the development and the operation of a camera-based data acquisition and processing system were identified. Based on these challenges and the initial research question, requirements were derived which led to the final specification of the system. Finally, the scope of the specification was reduced to the application of people detection and tracking. In the next chapter, fundamentals of object detection and tracking, which are needed to implement the Shopfloor Monitor, are explained.

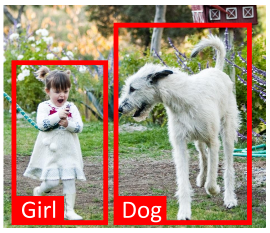
Chapter 3

Fundamentals

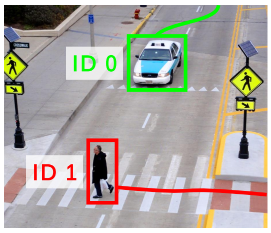
The following chapter briefly introduces the foundations of visual object detection and tracking, which are needed for the subsequent implementation of the Shopfloor Monitor. Both detection and tracking are well studied subfields of computer vision research and are used in many practical applications, such as face recognition, video surveillance, self driving cars and cashier-less stores. Both methods are characterized as follows:

Object Detection: The input image contains several objects of different classes. The task is to find bounding boxes for all distinct objects in the image and assign one class label out of a set of possible labels and a confidence score for this label to each object (see fig. 3.1a). [8–11]

Object Tracking: The input is a video, containing one or multiple moving objects. The task is to find bounding boxes around the objects in each frame and assign a constant ID to each object. Information about the object's appearance and motion from previous video frames is used (see fig. 3.1b). [12–15]



(a) Object detection.



(b) Object tracking.

Fig. 3.1: Examples of visual object detection and tracking. Object detection is used to localize and classify objects in an image. Object tracking locates objects in subsequent frames of a video, creates a trajectory and assigns a constant ID. (Background images from [16, 17])

In section 3.1 and section 3.2 functional pipelines for both object detection and tracking are explained. Section 3.3 highlights differences between object detection and tracking and explains how the two tasks complement each other. Based on this, section 3.4 explaines how both methods can be used in combination to achieve superior tracking accuracy and speed. Supplementary material, including a thorough literature overview of state-of-the-art methods for object detection and tracking, can be found in appendix A and appendix B.

3.1 Object Detection

The common framework of object detection algorithms is presented in fig. 3.2. The first step of the detection pipeline is the proposal of possible bounding box regions. This is done either by sliding a fixed-size window over a multi-scale image pyramid of the input image [18, 19], or by means of the selective search algorithm [20]. The latter is computationally less expensive and uses a clustering algorithm to iteratively merge proposed regions based on their similarity in color, texture, size and shape. After region proposal, robust features of every proposed region are extracted. Typically such features are obtained via Histograms of Oriented Gradients (HOG) [18], Scale-Invariant Feature Transform (SIFT) [21], as Haar features [22] or as deep convolutional features [23]. The extracted features are used in the next step to classify the object within each proposed bounding box. Bounding boxes with low-confidence labels are filtered out as they are likely to not contain any valid object. Features are also used for bounding box regression, whereby box dimensions and positions are modified to better fit the object. This is needed because the region proposal algorithm usually returns boxes of fixed sizes and aspect ratios which do not fit the object well. After this step, the image usually still contains a high number of largely overlapping bounding boxes because the proposal algorithm tends to propose multiple bounding boxes for the same object. Non-maximum-supression (NMS) [24–27] removes these duplicate bounding boxes and leaves only one box per object. The output of this detector pipeline is a set of bounding boxes containing one object each. Every box contains an according class label and a confidence score.

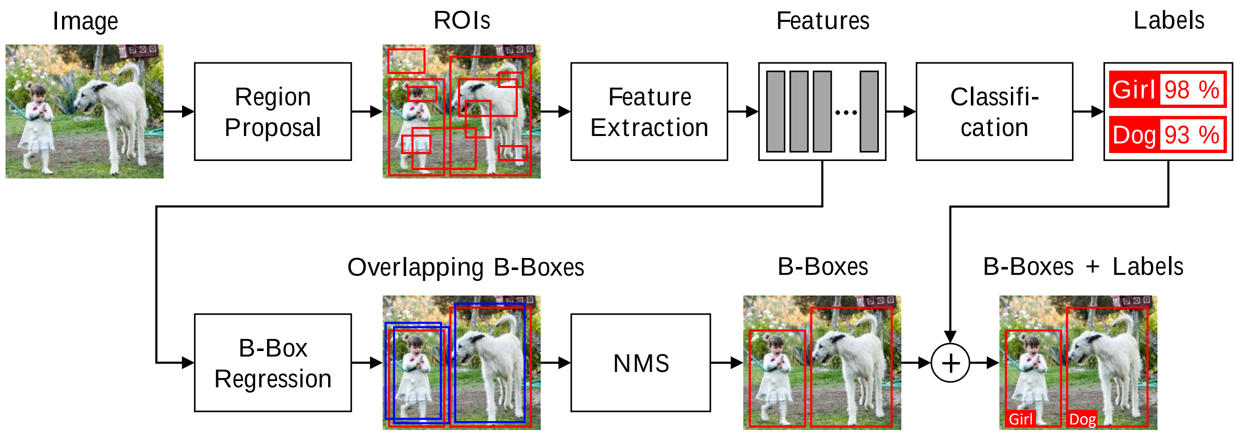


Fig. 3.2: Common pipeline of object detection algorithms. The actual implementation of each step varies for each individual algorithm. Newer models try to merge all steps in a single deep neural network to reduce computational cost and simplify training. [28] (Background image from [16])

3.2 Object Tracking

The scope of object tracking is to predict bounding boxes for all objects in a given video frame at time t_n based on the objects' motion and appearance in previous frames at times $t_0, t_1, \ldots, t_{n-1}$. The tracked objects are also referred to as *targets*. Tracking yields the trajectory of each target in a video sequence and preserves the identity of targets over time. The following explanation refers to single-target trackers [29, 30], but can easily be expanded to multi-target trackers [31–36]. A typical single-target tracker contains a *motion predictor*, a *feature extractor*, a *classifier* and a *model updater*, which have the following functions according to [37–39].

Motion Predictor: This module contains a motion model of the target. It predicts the most likely bounding box of the target in the next frame, based on the previous box dimension and box velocity.

Feature Extractor: The feature extractor extracts features from the image patch within the target's bounding box to represent the target in a robust way. These features are more robust to occlusion and changes of appearance, such as illumination or scale.

Classifier: The classifier compares features of the bounding boxes proposed by the motion model with a template of the target and determines which box most likely contains the target.

Model Updater: The model updater renews the target template based on the prediction result of the previous tracking cycle. This allows the tracker to account for appearance changes of the target throughout the video.

The typical object tracking pipeline is shown in fig. 3.3. Before being able to track a target, the tracker has to be initialized. This is done by providing the initial frame and the initial bounding box of the target, which is either created manually or found by an object detector. The feature extractor extracts features of the target's bounding box region, which serve as a template for the tracking target. This template is not constant, but updated by the model updater in subsequent cycles of the tracker. After initialization, the tracker receives a new frame and uses the motion model to predict candidate bounding boxes for the target. In the beginning, the motion model is inaccurate, but it becomes more accurate after a few frames. In the next step, a featureset is extracted from each candidate region and compared to the template features by the classifier. The bounding box with features most similar to the template is predicted as the new bounding box of the target. [37, 38, 40]

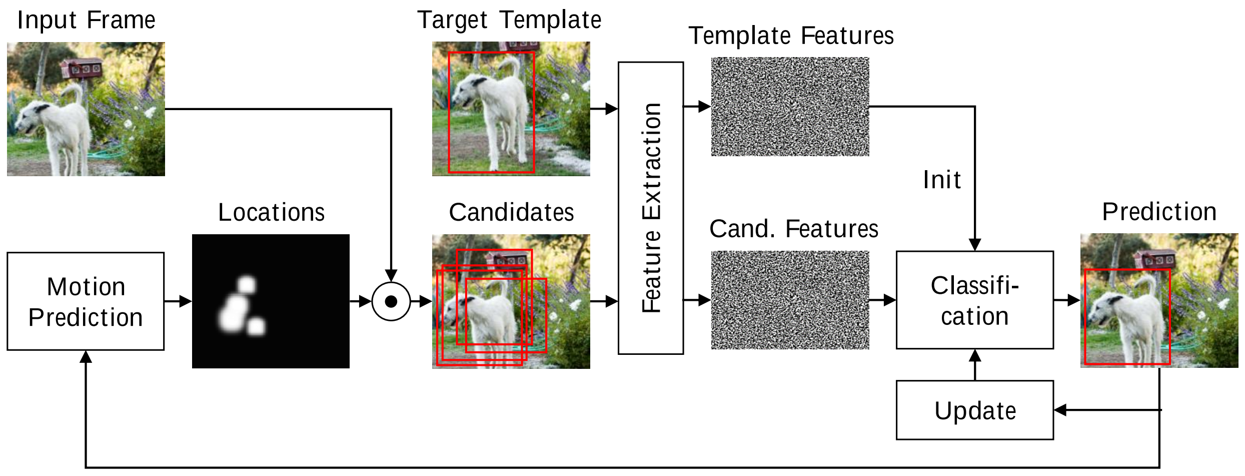


Fig. 3.3: Common pipeline of object tracking algorithms. The actual implementation of each step varies for each individual algorithm. Newer deep learning models merge some or even all of the individual steps in a single deep neural network to increase tracking accuracy. [37, 38, 41, 42] (Background image from [16])

Tracking multiple targets requires keeping track of the target identities, which is done by assigning a unique ID to each target. An additional data association step is needed to ensure that IDs stay constant. During this step proposed bounding boxes are assigned to existing target trajectories and IDs. Popular methods for data association are Joint Integrated Probabilistic Data Association (JIPDA) [43], Multiple Hypotheses Tracking (MHT) [44], Probability Hypothesis Density filters (PHD) [45] and Siamese neural networks [46–48]. Multi-target trackers can easily be realized by using an individual single-target tracker for every target. The trackers operate independently and without any cross-optimization [49].

3.3 Differences between Tracker and Detector

The different operational principles lead to complementary characteristics of object detectors and trackers. The object detector is trained offline on a large set of examples. Thus, it generalizes well and prevails at finding new targets in a frame. However, modern object detectors are complex, computationally expensive and they take a significant amount of time to find targets in a single video frame. Detection also does not perform any matching of targets across frames. It simply returns an unordered set of bounding boxes for each frame.

Object trackers are different as they utilize information about the target's motion and appearance in previous frames. This allows them to preserve the identity of targets across time. Under the assumption of small inter-frame motion, the tracker can perform a local search around the previous known location to find the target. This makes the tracker much faster than the detector. However, as the tracker is trained online on an individual target, it is less generalized and can not be used to find new target instances. Online training makes the tracker also prone to drift, as small localization errors accumulate over time (see appendix B.1). Thus, using only a tracker leads to poor accuracy of tracking results.

3.4 Combining Tracker and Detector

To leverage both speed of the tracker and high accuracy of the detector, it is best to combine both methods. This is called *tracking-by-detection* [50–53]. The first method is to run the detector on every video frame and use a tracker only for data association. Detections are matched across frames, preserving identity of targets and forming trajectories. Methods used for matching are the Hungarian method [54], linear programming [55, 56], k-shortest path optimization [32] or Siamese neural networks [57]. However, running the detector on every frame is computationally expensive and slow.

A better method is to run the detector only on every nth frame and use a tracker to predict target bounding boxes in the $n-1$ intermediate frames [40, 58]. After every nth frame detections are matched to an existing trajectory, and the according tracker is reinitialized based on the detection. This prevents tracker drift as errors do not have time to accumulate. Because of the lower computational complexity of the tracker, this approach is faster than running the detector in every frame. The Shopfloor Monitor uses this approach. Instead of running the detector in fixed intervals, it can also be scheduled adaptively based on the tracking quality [59].

3.5 Conclusion

This chapter briefly introduced visual object detection and tracking. After a high-level explanation, a functional pipeline of each method was described. These pipelines contained the basic building blocks and processing steps, found in most detection and tracking algorithms. They helped to build a general understanding of both tasks without going too much into detail of individual algorithms (see appendix A and appendix B). Afterwards, the differences between object detection and tracking were highlighted. This clarified how both methods complement each other and can be used in combination to achieve both high accuracy and speed. The foundations presented in this chapter are needed to understand the implementation details of the Shopfloor Monitor in chapter 5. However, before this, publications of tracking and visualization solutions similar to the Shopfloor Monitor are reviewed in the next chapter.

Chapter 4

Related Work

This chapter provides a literature overview of existing solutions for multi-target tracking and visualization in multi-camera environments, which are related to the Shopfloor Monitor. Research in this field is a sub-topic of multi-camera multi-target tracking [60, 61]. First, the general concepts of these and related solutions are explained and the scope of the literature review is defined. Afterwards, a framework common to all tracking systems is presented. The subsequent analysis is oriented at this framework and explains, how the different solutions implement various aspects of the framework. In the end of this chapter, limitations of the presented solutions and further research directions for this work are identified.

4.1 Multi-Target Tracking Solutions

Aim of the analyzed multi-camera multi-target tracking solutions is to capture a scene containing multiple moving targets, such as humans or objects, with multiple cameras simultaneously and reconstruct this scene in real-time within a virtual 2D or 3D model. The virtual model condenses the information carried within a large number of simultaneously acquired video feeds. It lets the user experience the captured scene more naturally and prevents him from missing important events. Thus, the presented tracking solutions improve the limited capabilities of classical multi-camera visualizations, which simply show each acquired video feed on a separate monitor. Prior work has been done to improve these classical solutions, so as to enhance the situational awareness and reduce the cognitive strain of the observer [62–70]. However, these solutions still follow a more classical approach and do not utilize a virtual model to represent the camera feeds. Thus, the following analysis focuses on solutions, which reconstruct the observed scene and tracking targets within a virtual model [71–81]. A similar class of tracking solutions embed the raw video streams as textures into the 3D model [82–90]. However, as these solutions are less similar to the Shopfloor Monitor, they are not covered in this review. Similarly, this review excludes publications related to multi-camera multi-target tracking, which focus exclusively on the detection and tracking subsystem, without implementing any visualization of the tracking results [91–98].

4.1.1 General Framework

Most of the tracking solutions implement the same basic framework. It has a high similarity to the framework of the Shopfloor Monitor presented in fig. 2.1. Video streams are acquired from multiple synchronized cameras, which may or may not have overlapping fields of view. Object detection is used in conjuncture with multi-target tracking to localize people and objects within each new frame of the video streams and assign them to an existing trajectory. Targets are then visualized in a virtual model of the real scene. To map from image to world coordinates, either the camera pose or a homography matrix is estimated, based on point correspondences between the camera images and the real world. Apart from these basic functions, some of the solutions incorporate more advanced features, such as face recognition [71], activity recognition [72, 73, 77], object classification or human motion analysis [72].

4.1.2 Camera Types

The analyzed literature utilizes a broad range of cameras types. Most commonly used are multiple static cameras [71, 74, 77], which have a constant pose and focal length. Thus, they have to be set up and calibrated only once, and enable the use of background subtraction as simple object detection method. The cameras have either overlapping fields of view (FOV) as in [77] or separate FOVs as in [71]. All approaches use monocular cameras except for the one presented in [81], which makes use of multiple stereo-cameras to track an object in three dimensions. The cameras can be either wide-angle cameras, which use a lens with a small focal length, as in [76], or a camera with a normal lens as in [74, 78]. Instead of using multiple equivalent cameras, the VSAM tracking system [72, 99–101] comprises different camera types, such as five statically mounted cameras, a hemisperical camera, two mobile cameras mounted on a van and an aerial drone and several indoor surveillance cameras. Another execption is the system developed in [78]. It uses only a single static camera instead of multiple camera, as as all other solutions do. Some of the systems also use pan-tilt-zoom (PTZ) cameras instead of statically mounted cameras [75, 76]. PTZ cameras allow for dynamic adjustment of the camera pose and the focal length. They can be used in a master-slave configuration, where one wide-angle camera, the master, observes the whole scene and controls several PTZ cameras, the slaves. These slaves follow a target and capture close-up shots of it [76, 100]. The last difference between the cameras used, refers to the level of integration of processing components. Usually, simple IP cameras are used and all visual processing tasks are executed on either a centralized PC or an individual PC for each camera. However, some works utilize smart cameras, which have an embedded field-programmable gate array (FPGA), digital signal processor (DSP) or PC, on which the video processing is performed [73, 74].

4.1.3 Detection and Tracking

Object detection is mostly realized via simple background subtraction. Here, a statistical background model is created and subtracted from each frame, leaving only interesting foreground pixels, which correspond to the moving targets. This is possible, because the cameras are static and, thus, the background stays unchanged between subsequent frames. The bounding box of extracted foreground pixels is computed and a point within this box used to represent the target [74, 77, 78]. Instead of background subtraction [78] extracts HOG cascades with a sliding window and classifies them into target and background via a state vector machine (SVM), and [76] uses motion history image (MHI) [102] for combined detection and tracking.

Detected targets are then tracked either by multiple individual single-target trackers [74] or by a central multi-target tracker [77, 79], which performs cross-optimization between targets. Tracking is performed in two steps. The first is the association of detections to existing tracks by algorithms, such as JPDAF [77], correlation matching [101] or the auction algorithm [78]. Matching is performed based on spatial proximity and appearance features. The second step is the prediction of the future target location based on a simple motion model and a Kalman filter [78] or particle filter [74, 77]. While tracking is usually performed in image coordinates, the tool developed in [77], first maps target locations into world coordinates and then performs tracking in world coordinates. The DOTS system [71, 103–106] improves tracking through occlusion by explicitly handling split and merge scenarios of multiple targets. Most tools perform target handover between adjacent camera views by extracting and matching appearance features upon redetection of the same target in a new camera view [71, 101]. Moreover, redundant views of the same target are merged to get a consistent representation of each target in world coordinates [74]. The system presented in [73] goes even further and uses logical reasoning about interactions of targets with the environment to generate a consistent representation of the target in the virtual model.

4.1.4 Coordinate Mapping and Camera Calibration

Mapping target locations from image coordinates to the virtual model is usually achieved by means of the intrinsic and extrinsic camera parameters [73–75, 79, 101, 103], which are estimated via camera calibration in a prior step. During calibration, known point correspondences between the image and the virtual model are used to estimate the camera parameters. Some solutions provide auxiliary tools for this calibration step [71, 72, 74]. The tracking systems using PTZ cameras [75, 76] have to update the extrinsic camera parameters after every new camera adjustment. Since the adjustment is known, new parameters can be computed automatically and no manual re-calibration is necessary. Instead of using extrinsic camera parameters, other solutions use a homography matrix to describe the perspective transformation between a camera image and the planar ground-plane of the virtual model [77, 78, 80, 107]. Similar to the extrinsic camera parameters, the homography is estimated from point correspondences. The planar ground-plane assumption is made by almost all tracking systems to enable computation of the target locations on this plane from a single camera view. Without this assumption, the target must be visible in at least two camera views simultaneously at every time. A special solution for camera calibration is proposed in [76]. Because, here, the virtual model is built directly from images of the cameras in the system, the extrinsic camera parameters are known, superseding the need for a separate calibration procedure.

4.1.5 Creation of the Virtual Scene Model

The analyzed tracking systems vary greatly in terms of how the virtual model of the observed scene is created. This is either done manually via computer-aided design (CAD) software [77, 103] or via 3D rendering engines, such as OpenGL [81] or the Unity 3D game engine [73]. Alternatively, the 3D model can be photogrammetically reconstructed from multiple images of the scene with different methods. For example, [79] uses Structure-from-Motion [108], and [78] utilizes city dense modeling [109] in combination with a 3D CAD model. Both methods require previous acquisition of multiple images of the scene via external cameras. The approach in [74] works similarly, however makes use of three additional laser scanners and an attitude sensor to register image positions and orientations in the world coordinate system. The method for photogrammetric reconstruction in [76] differs from the previous solutions, as it uses the native system cameras instead of an external camera to acquire images. Instead of manually creating a virtual model, the solutions in [72, 75, 80, 107] use available models from Geographic Information Systems (GIS), such as Google Earth. Finally, DOTS [71] provides not only a virtual 3D model, but also displays tracking targets and camera FOVs in a 2D floorplan.

Some of the tracking systems not only visualize the current location and past trajectories of the targets, but incorporate additional functions. For example, [72] embeds a raw video of an aerial view acquired from a drone into the virtual model to provide an additional overview of the scene. The simulation tool in [78] uses heat maps as wall textures to visualize how frequent pedestrians look at a certain area of the wall. This is used to determine effective positions for advertisements. The virtual model in [77] provides advanced weather and lighting animations, such as sunlight, rain and fog as well as day and night cycles. These animations are synchronized with the prevalent conditions to enhance conformity of the model with the real scene. This makes it more tangible for the observer.

4.1.6 Target Representation in the Virtual Model

All tracking solutions represent the tracking targets within the virtual model either as a simple geometric shape [71, 75], as an animated *virtual actor*, or as a live texture. The virtual actor concept is implemented in [72, 73, 77] and requires a module for recognition of the activities performed by the targets. Live textures are extracted foreground pixels of each target, which are mapped onto a planar box in the virtual model, located at the predicted target location. This representation is implemented in [74, 76, 80, 107]. The tracking software presented in [79] provides both a simple symbolic representation and live textures. The tool in [78] is an exception, as it does not represent the current targets in real-time as all other solution do, but instead visualizes only historic trajectories.

4.1.7 User Interaction with the Virtual Model

An important aspect of the virtual model is the interaction with the human user, as it determines how much information the user can obtain from it. As the observed scene is abstracted from the raw video streams and reconstructed in an immersive virtual model, it is possible to re-render the scene from any desired viewpoint. This viewpoint does not need to coincide with any of the original camera views. Some of the solutions allow for arbitrary movement with 6 degrees of freedom through the model [71, 74, 75, 77, 80, 107]. Others allow the user to place a *virtual agent* into the model and observe the scene from his viewpoint [72, 79]. The solutions presented in [74, 77, 79] additionally visualize the scene from the viewpoint of any of the tracking targets. A slightly simpler visualization mode with predefined zoom levels and viewing positions and angles is described in [81]. Apart from live-monitoring of the scene, most tracking systems also allow for replay of past content. This enables the user to navigate forward and backward in time. The DOTS system [71] takes this concept further and provides a non-linear timeline with markers indicating automatically detected events. The user can additionally drag objects on the 2D floorplan or in the raw videos along their trajecoties to navigate through time [106].

4.2 Limitations of the Existing Solutions

The presented solutions are fully functional multi-camera multi-target tracking solutions and some of them, such as DOTS [71] and VSAM [72] were developed over the course of many years, thus being mature, market-ready systems with a large number of features. Some of those system are even patented [110, 111]. Thus, it is not surprising, that research in this field has stagnated and 15 out of the 19 analyzed solutions were published before 2012. Due to this, most of the presented solutions utilize detection and tracking algorithms, which lag behind the state of the art significantly. Even in the four recent publication from 2015 to 2017, object detection is performed with the outdated background subtraction approach. To advance research in this field, the present work uses state-of-the-art deep learning models for object detection and fast correlation filter-based object trackers. A methodological issue with some of the works [73–76, 78–80, 107] is the missing evaluation of the tracking performance. To provide a better understanding of the detection and tracking system, an in-depth analysis of the tracking performance shall be part of this work. The biggest issue of the presented solutions is their limited scope of applications. All solutions aim at summarizing multiple parallel video streams into the condensed representation of a virtual 2D or 3D model, which is then simply observed by the user for surveillance. None of the publications explores further applications of the created virtual model. Thus, this work aims at providing the infrastructure for the much broader range of applications presented in chapter 2.

4.3 Conclusion

In this chapter a literature review of full-fledged multi-camera multi-target tracking solutions was conducted. Hereby, the focus was narrowed down to a small subset of only 19 publications, which presented entire software solutions, considering not only object detection and tracking, but also camera calibration, coordinate transformation and visualization. Common to all publications was the reconstruction of the observed scene in form of a virtual 2D or 3D model. This model visualized tracking targets in real-time and allowed unrestricted observation of the scene by the user. At the end of the chapter limitations of the presented solutions were described and suitable research directions for this work were identified to advance the state of the art. The next chapter covers the implementation of the Shopfloor Monitor.

Chapter 5

System Implementation

The aim of this chapter is to describe the realization of the Shopfloor Monitor and its components in a top-down fashion. It starts with an overview of the system architecture and subsequently breaks it down into smaller sub-systems, which are then explained in detail. Apart from describing the software, additional emphasis is laid on the camera calibration.

First, section 5.1 gives a high-level overview of the system architecture and its modules. Section 5.2 expands on the main server process, which implements the majority of the system functions, such as data input and output as well as the tracking of targets. The detection and tracking framework as the core component of the Shopfloor Monitor and part of the main process is explained in depth in section 5.3. Subsequently, section 5.4 describes the MySQL database used for long-term storage of tracking results. Section 5.5 goes into detail about the web app framework for building client apps, which make use of the generated tracking data. It introduces an app for real-time visualization of target locations and trajectories on a 2D floorplan of the factory. In section 5.6 the cameras used in the prototype of the Shopfloor Monitor are set up, and subsequently their calibration is described in section 5.7. Finally, section 5.8 critically reviews issues of the current realization of the Shopfloor Monitor and suggests future improvements.

5.1 Full System Architecture

The Shopfloor Monitor consists of three major modules, a pool of arbitrary many IP cameras, an on-site analysis PC and a remote client. Details about the hardware specification and the software environment of the analysis PC can be found in appendix C.1. The overall architecture of the Shopfloor Monitor is shown in fig. 5.1. All IP cameras and the analysis PC are required to be part of the same network, which can be either the Internet or a local network. This way, camera streams can be acquired by the main process on the analysis PC via real-time streaming protocol (RTSP) (see section 5.6). The main process also executes algorithms for detection and tracking of humans and sends results to the MySQL database server [112] and to the web apps running on a remote client. This client can be any device with a web browser and represents the interactive user interface. Web apps are served over the Internet by an Eventlet [113] webserver running within a Python Flask [114] app. Client web apps might be used, for example, to visualize or analyze tracking data.

Although being suitable as a long term storage for tracking results, the MySQL database is not sufficient for low-latency communication between the main process and the client web apps. Thus, websockets based on Flask-SocketIO [115], which are managed by the Flask app, are used to pass messages between the main process and the client. These messages contain, for example, the world positions and IDs of all tracking targets. They are sent with a frequency of approximately 15 Hz and, upon receipt, trigger an event in the client for updating the visualization with the new tracking data. Other messages notify the client when the main process has terminated.

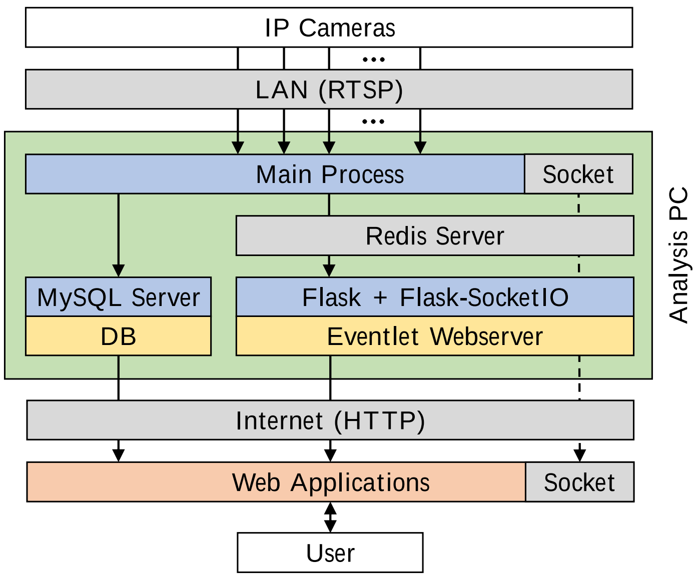


Fig. 5.1: Implementation-oriented architecture overview of the Shopfloor Monitor. Camera streams are acquired over the network via the local analysis PC to detect and track people (main process). A MySQL database and websockets are used to communicate the results to the remote client apps, which are served by an Eventlet webserver running within a Python Flask app.

Even though the Flask app could be integrated directly into the main process, a much cleaner and easier to maintain solution is executing it in a separate process. A Redis messages broker [116] is then used to realize inter-process communication between the main process as well as the Flask app. This enables both processes to be run completely decoupled, while maintaining their ability to act as servers. Thus, data can be passed on to the client apps from either of the two processes.

Websockets allow for bidirectional data transfer between the main process or the Flask app and the client apps. Hence, they can be used not only for passing tracking data from the two server processes to the client, but also for passing user inputs in the opposite direction. This can be useful, if a client app does not only passively visualize data, but allows for user inputs which affect execution of the main process or the Flask app.

The combination of an SQL database and websockets provides a flexible interface for the development of arbitrarily complex web apps. These apps enable interaction with the resulting tracking data of the Shopfloor Monitor. On one hand, the database stores this data in the long term and enables web apps to retrieve historic information about targets and trajectories, which can be used for analysis purposes. On the other hand, websockets enable fast, event-based communication between the main process or Flask app and the client web apps. When developing new client apps, the existing socket interface can easily be reused, and extended with additional events to handle user-interaction and data visualization.

5.2 Main Process

The main process on the analysis PC is the most important component of the Shopfloor Monitor. It acquires the camera streams, runs the detection and tracking algorithms and communicates with the client web apps and the database. Figure 5.2 shows the architecture of the main process, which comprises a number of different threads. These threads can be executed virtually parallel. Each of the N cameras $C_1, C_2, \dots, C_N$ is assigned to an individual acquisition thread which captures and resizes every new frame F_j, provided by the camera C_j at the discrete time step $t > 0$, in an infinite loop. A threading barrier is used to synchronize the acquisition threads. This guarantees that all acquisition threads have captured their according frame at time step t before capturing the next frame for $t+1$. In case any camera fails to capture the next frame, the program raises an error message and halts. Each successfully captured frame F_j is inserted into a first-in-first-out-queue (FIFO-queue) Q_j, which is associated with the jth acquisition thread and capable of storing at most 128 subsequent frames. As these queues are thread-safe, stored frames can be safely read from within the main thread for further processing. This is done in an infinite loop, which first collects frames F_j of all cameras at a certain time step t and aggregates them in a single array $F = (F_1, F_2, \dots, F_N)$. This frame array is then fed into the detection and tracking framework, which outputs the data package $\mathcal{D}$ containing, inter alia, target locations in world coordinates, associated target IDs and bounding boxes in image coordinates. Generated data packages $\mathcal{D}$ are then inserted into another FIFO-queue $Q_{\mathcal{D}}$ for transfer to the output thread. Here, data packages are first read from the queue $Q_{\mathcal{D}}$, and then written to the logfile, inserted into the MySQL database and sent to the client web apps via websockets.

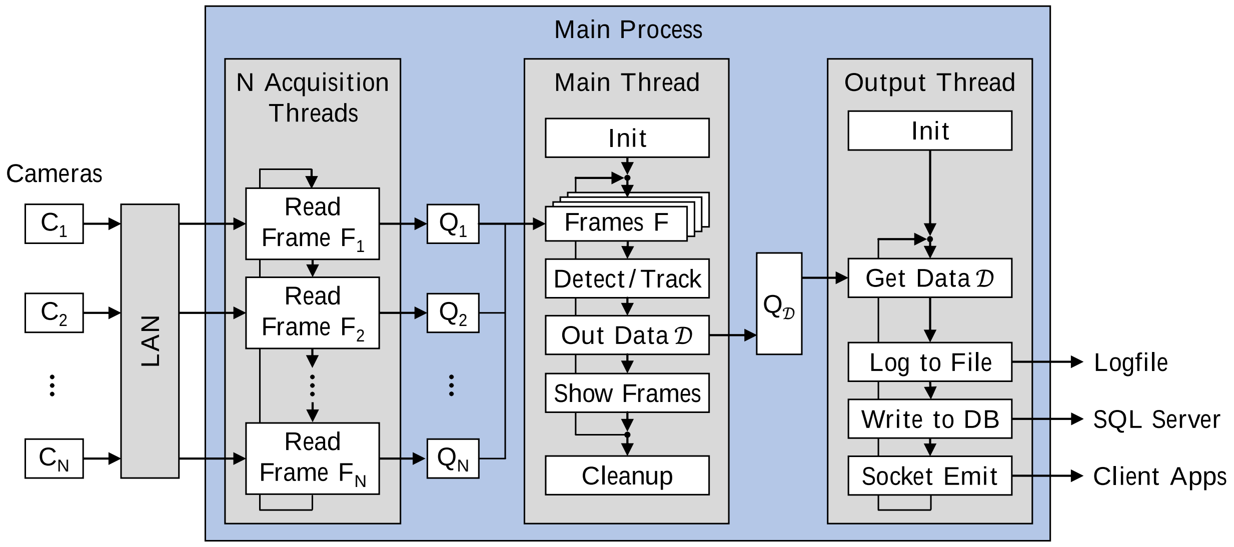
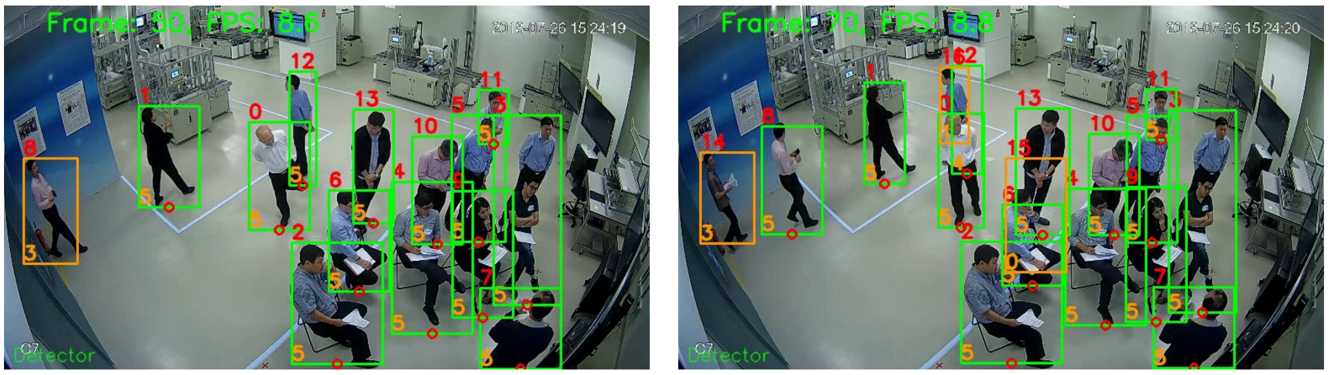


Fig. 5.2: Architecture of the main process. This process simultaneously acquires N parallel camera streams, detects and tracks people, and visualizes and broadcasts tracking results.

The output queue $Q_{\mathcal{D}}$ serves as a buffer, which allows for a relatively constant output rate despite significantly varying cycle times of the main thread. Maintaining a constant output rate requires the output queue to always contain enough elements to bridge delays in the supply of new output data packages $\mathcal{D}$. To achieve this, the output thread employs an adaptive strategy for polling of the output queue. By choosing the delay between two subsequent reads to be inversely proportional to the momentary length of the output queue and by keeping at least 15 elements in the queue at all times, a relatively constant length of the output queue, and thus an output rate of approximately 15 Hz, can be ensured.

In addition to computing and transferring tracking results $\mathcal{D}$ to the output thread, the main thread directly draws some of the tracking results, such as bounding boxes and box IDs, onto the according frames and displays, respectively stores them for debugging purposes. Two example output frames are shown in fig. 5.3a. This output also enables the user to safely stop the main process with a keyboard shortcut. Upon receipt of this shortcut a cleanup procedure is initiated, which terminates the output and acquisition threads, garbage-collects remaining queue elements and closes camera connections. Part of the main thread is also an init procedure, which is run once at the start of the main process. Here, command line arguments are parsed, the camera connections are established and calibration parameters are loaded. Furthermore, output and acquisition threads are started, queues are created and the detection and tracking framework is initialized. Finally, it is ensured, that the Redis and MySQL servers are running.

Similarly to this, the output thread also has an init procedure, in which the logging directory is created, if not already existent, and the database and socket connections are established.



(a) Tracker output frame at step $n = 50$.

(b) Tracker output frame at step $n = 70$.

Fig. 5.3: Visualization of the tracking results in an output video stream. Every tracking target has a bounding box in either green, if the target is valid, or orange, in case of an invalid target. Red numbers are box IDs and orange numbers are the current value of the target's match counter (see section 5.3.2.3 for an explanation). The foot point of each target is marked with a red circle. Additionally, the step number and the momentary frame rate are shown in the top left corner. The *detector* label in the bottom left corner indicates, that the detector is run on this frame and the trackers are updated with the detection results.

5.3 Detection and Tracking Framework

At the core of the Shopfloor Monitor's main process is the detection and tracking framework, which utilizes an object detector (see section 3.1) and multiple single-target trackers (see section 3.2) to locate people on the factory floor and estimate their trajectories over time. The main reason for combining the trackers with an object detector is to leverage both the high accuracy of the object detector and the high processing speed of the object tracker (see section 3.4). The trackers are updated with detections in regular intervals to prevent drift by accumulation of tracking errors. Using only an object detector is insufficient as the detector is too slow to be run on every frame. The detector also generates false positives and false negatives, which have to be filtered out. Furthermore, trackers are needed to preserve the identity of targets across subsequent frames. The framework developed here is inspired by the Simple Online and Real-time Tracking (SORT) system [117] and its improved version called DeepSORT [118]. The following list summarizes the most important features of the detection and tracking framework:

- Detection of multiple targets every f_d time steps and tracking of targets in intermediate time steps.
- Matching of detections and tracks via the Hungarian method applied to IoU of bounding boxes.
- State update of trackers based on detection results every f_d steps.
- Low-pass filter for dampening the impact of false positive and false negative detections.
- Transformation of target locations from image to world coordinates.
- Assignment of a unique ID to each tracking target.

Each of these aspects is explained in more detail in the following section. However, before this, the composition of the input and output data of the detection and tracking module, as well as its integration into the main thread, are further clarified. The module takes a frame array F, which is composed of all

Tab. 5.1: Composition of the output data package $\mathcal{D}$ of the detection and tracking module for a set of N input frames F. Note the difference between the number K of both valid and invalid targets and the number $\widehat{K}$ of only valid targets.

Variable	Symbol	Dimension	Description
Timestamp	t	1×1	Unix time at the moment of the assembly of the data package in the main thread.
Step Number	n	1×1	Number of discrete time steps (processed frame packets F) since start of the main process.
Num Cameras	N	1×1	Number of cameras in the Shopfloor Monitor. Equivalent to the number of frames in F.
Num Targets	K	$N \times 1$	Number of all tracking targets (valid and invalid) in each of the N frames in F.
Bounding Boxes	B	$N \times K \times 4$	Bounding box coordinates $(u_{\min}, \nu_{\min}, w, h)$ for each of the K targets within each of the N frames in F.
Box IDs	I	$N \times K \times 1$	Identifier number for each bounding box, which is unique within the same frame, but not unique across different frames.
Box Valid States	V	$N \times K \times 1$	Boolean value for each bounding box, indicating whether a target is valid or not.
Target IDs	T	$\widehat{K} \times 1$	Identifier for each valid target, which is unique across all frames.
World Positions	W	$\widehat{K} \times 2$	Position (X, Y) of each valid target on the ground plane of the factory in world coordinates.

camera frames $F_1, F_2, \dots, F_N$ acquired at the same time t, as an input, and outputs a data package $\mathcal{D}$ after each time step. This package contains tracking results, as specified in table 5.1.

Algorithm 1 shows the basic framework of the detection and tracking module and illustrates its integration into the main thread within the main process (see fig. 5.2) of the Shopfloor Monitor (see fig. 5.1). Requirements for running the module are the frame queues $Q = (Q_1, Q_2, \dots, Q_N)$, which buffer the input frames of all cameras, and the output queue $Q_{\mathcal{D}}$ for buffering generated tracking data packages $\mathcal{D}$ for a time step t. Additionally, the number of cameras N and the detector cycle frequency f_d, which determines how often the object detector is run, have to be specified. Initialization of the object detector and the tracker requires several additional hyperparameters to be set. These are the detection algorithm $\mathcal{A}_{\text{det}}$, the tracking algorithm $\mathcal{A}_{\text{trk}}$, the match IoU threshold θ_{IoU} and the hypothesis valid and deletion thresholds θ_v and θ_d. The meaning of these parameters is explained in the course of this section. For transformation of image to world coordinates the tracker needs the calibration parameters $P = (P_1, P_2, \dots, P_N)$ for each of the N cameras. This includes the camera matrix M, the new camera matrix M', the distortion coefficients k_1, k_2, k_3 and k_4, the scaling factor s and the homography matrix H. Computation of these parameters is explained later in this chapter. After creation of the detector and tracker instances with the according hyperparameters, the main thread enters an infinite loop, in which, first, the frame array F is assembled by reading the oldest frame F_j from each frame queue $Q_j \in Q$, and, second, the new target locations within the frames of F are estimated via the PREDICT method of the tracker. In addition to the location prediction, the detector is run every f_d steps yielding new target locations $\mathcal{D}'$. These locations are then used to update the trackers via the UPDATE method of the tracker. At the end of each main loop iteration, the results of the tracking step are retrieved, assembled into the output package $\mathcal{D}$ and put into the output queue $Q_{\mathcal{D}}$. The following section 5.3.1 explains the detection of targets in more detail, while section 5.3.2 elucidates the tracking, including the location prediction and the detection-based state update of the trackers.

Algorithm 1: Basic detection and tracking framework.

```
Data: $Q$, $Q_{\mathcal{D}}$, $f_d$, $N$
Data: $\mathcal{A}_{\text{det}}$, $\mathcal{A}_{\text{trk}}$, $P$, $\theta_{\text{IoU}}$, $\theta_v$, $\theta_d$
                                        ▷ Create detector and tracker and set hyperparameters
  detector ← new detector($\mathcal{A}_{\text{det}}$)
  tracker ← new tracker($\mathcal{A}_{\text{trk}}$, $P$, $\theta_{\text{IoU}}$, $\theta_v$, $\theta_d$)
  cycle_counter ← 0
  $n \leftarrow 0$
  while true do
                                        ▷ Get frames of all cameras and assemble packet $F$
      $F \leftarrow [\,]_N$
      for each $j \in \{1, 2, \ldots, N\}$ do
          $F_j \leftarrow Q_j$.POP()
      end for
                                        ▷ Update tracker based on detections every $f_d$ cycles
      if cycle_counter == $f_d$ then
          cycle_counter ← 0
          $\mathcal{D}' \leftarrow$ detector.DETECT($F$)                          (sec. 5.3.1)
          tracker.UPDATE($F$, $\mathcal{D}'$)                                   (sec. 5.3.2.3)
                                        ▷ Predict target locations in intermediate cycles
      else
          tracker.PREDICT($F$)                                            (sec. 5.3.2.2)
      end if
      cycle_counter ← cycle_counter +1
      $n \leftarrow n + 1$
                                        ▷ Retrieve tracking results (sec. 5.3.2.4)
      $W, T \leftarrow$ tracker.GET_TARGETS()
      $K, B, I, V \leftarrow$ tracker.GET_STATE_INFO()
                                        ▷ Assemble output package and put it in output queue
      $\mathcal{D} \leftarrow (t, n, N, K, B, I, V, T, W)$
      $Q_{\mathcal{D}}$.PUT($\mathcal{D}$)
  end while
```

5.3.1 Detection

The object detector can be any algorithm which takes an array of frames F as input and outputs a set of bounding boxes for the objects in each frame. Potentially useful detectors are introduced in appendix A. In particular, the current implementation of the Shopfloor Monitor uses the detectors included in the Tensorflow Object Detection library [119]. This is because they share a common interface and, thus, can be easily interchanged to enable usage of the detection algorithm which best meets the requirements. Apart from returning the bounding box coordinates B' of each object, the detector returns the number K' of detected objects per frame as well as the classes C' and confidence scores S' of all objects (see table 5.2). Objects with confidence levels $S' < 0.3$ are discarded.

When instantiating the object detector, a frozen Tensorflow inference graph is loaded from its corresponding file and de-serialized into a new compute graph in the memory of the graphics processing unit (GPU). In addition to the model architecture, the frozen inference graph contains the pretrained weights of the object detector. By calling the DETECT routine of the detector on an array of frames F, each individual frame in F is fed into the compute graph for inference and the detection outputs are accumulated in $\mathcal{D}'$.

Tab. 5.2: Composition of the detector output $\mathcal{D}'$ for a set of N input frames F.

Variable	**Symbol**	**Dimension**	**Description**
Num Detections	K'	$N \times 1$	Number of detected objects in each of the N frames in F.
Bounding Boxes	B'	$N \times K' \times 4$	Bounding box coordinates $(u_{\text{min}}, \nu_{\text{min}}, w, h)$ for each of the K' detected objects within each of the N frames in F.
Object Classes	C'	$N \times K' \times 1$	Categorical class label, specifying the category of each detected object.
Confidence Scores	S'	$N \times K' \times 1$	Confidence level of the class label of each detected object in percent.

Inference is computed on a single GeForce GTX 1080 TI GPU (see appendix C.1).

Since the object detectors are trained on Microsoft's Common Object in Context (COCO) dataset rather than an application-specific dataset, the detector has to constantly generalize to unknown data when being used in the Shopfloor Monitor. However, due to the lack of an application-specific training dataset re-training of the detector is not possible, and thus a slightly increased number of faulty detections have to be accepted. One of the most salient and frequent detection faults are oversized bounding boxes, which fill almost the entire frame. Assuming that people in the observed scene can not exceed a specific size on the frame, it is easy to filter out such oversized boxes by limiting the maximum allowed box size to 25 % of the frame width and 60 % of the frame height.

In this work, the Shopfloor Monitor tracks only human targets. To achieve this, the object detector has to be retrained on a ground-truth dataset containing only humans. Again, no such dataset is available, which is why, instead of retraining, a simple filter is employed to remove all detections with predicted class labels other than *person*. If this filter is removed, the object detector can distinguish between 80 different object categories making the Shopfloor Monitor applicable to other use cases.

5.3.2 Tracking

Task of the tracker is to predict the bounding boxes B of each target in a new set of frames F at time step t based on the location of bounding boxes in previous frames at time step $t-1$ (see section 3.2). Each target is assigned to its own single-target tracker, the *core tracker*, which is used to estimate the location of this particular target in subsequent frames. No cross-optimization between the core trackers of individual targets is performed to keep complexity of the tracking framework low. Note that the term *tracker* is used for the ensemble of the core trackers of all targets in the frames F, whereas the term *core tracker* refers to the single-target tracker associated with a sole target. In the detection and tracking framework the OpenCV implementations of the Minimum Output Sum of Squared Error (MOSSE) tracker, Kernelized Correlation Filter (KCF) tracker and Discriminative Correlation Filter with Channel and Spatial Reliability (CSRT) tracker are used as core trackers, but can be replaced with any other tracking algorithm as long as it has a compatible interface. As learned from algorithm 1, the tracker has two different operational modes. The first is the prediction mode, in which the locations of known targets in a new set of frames F are estimated based on the previous target locations. The second is the update mode, in which the core trackers are updated based on the results of the object detector. Prediction and update mode are entered based on whether the PREDICT or UPDATE method of the tracker is called.

The UML-diagram in fig. 5.4 shows the attributes and methods of the tracker, which is explained in more detail in the following sections. Apart from the already explained PREDICT and UPDATE methods, the tracker has a constructor for setup of constants, such as the tracking algorithm $\mathcal{A}_{\text{trk}}$, the calibration parameters P, and the thresholds θ_{IoU}, θ_v and θ_d. The INITIALIZATION method initializes all remaining attributes of the tracker. GET_STATE_INFO is used to retrieve the current values of the internal tracker attributes, and GET_TARGETS computes world positions W and unique target IDs T of the currently tracked targets. The following sections explain the important methods of the tracker in more detail.

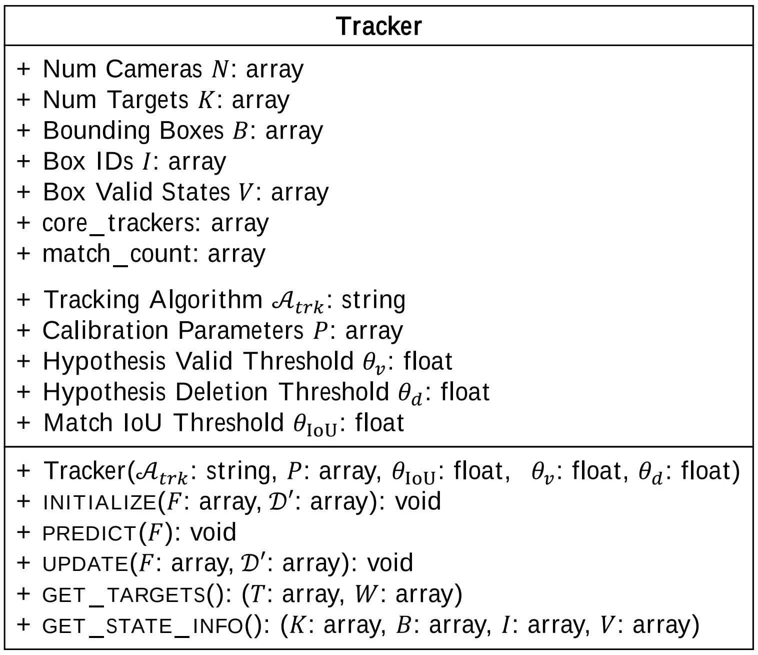

Tracker
+ Num Cameras N: array
+ Num Targets K: array
+ Bounding Boxes B: array
+ Box IDs I: array
+ Box Valid States V: array
+ core_trackers: array
+ match_count: array
+ Tracking Algorithm $\mathcal{A}_{trk}$: string
+ Calibration Parameters P: array
+ Hypothesis Valid Threshold θ_v: float
+ Hypothesis Deletion Threshold θ_d: float
+ Match IoU Threshold θ_{IoU}: float
+ Tracker($\mathcal{A}_{trk}$: string, P: array, θ_{IoU}: float, θ_v: float, θ_d: float)
+ INITIALIZE(F: array, $\mathcal{D}'$: array): void
+ PREDICT(F): void
+ UPDATE(F: array, $\mathcal{D}'$: array): void
+ GET_TARGETS(): (T: array, W: array)
+ GET_STATE_INFO(): (K: array, B: array, I: array, V: array)

Fig. 5.4: UML-diagram of the tracker, used as part of the detection and tracking framework.

5.3.2.1 Tracker Initialization

Before being able to predict locations of the targets, the tracker needs to be initialized with new tracking targets at time step $t = 0$. These targets are taken from the results $\mathcal{D}'$ of an initial detector run. As shown in more detail in algorithm 2, first all attributes of the tracker, such as the number of cameras N, the numbers of targets K as well as the array of bounding boxes B, box IDs I and box valid states V are inferred from the frame array F as well as the initial detection outputs B' and K'. Initially, all bounding boxes are set to the invalid state, as they are only validated on multiple successful redetections in later update steps. After this, a new core tracker is created and initialized for every bounding box in B in the frames F at time step $t = 0$. The hyperparameter $\mathcal{A}_{\mathrm{trk}}$ determines, whether MOSSE, KCF or CSRT is used as tracking algorithm. Each core tracker has a related match counter, which is needed later. It is initialized to 1. The notation $[\;]_{N \times K}$ stands for the initialization of an empty array with N subarrays of length K_j for each $j = 1, 2, \ldots, N$.

In the detection and tracking framework (see algorithm 1), tracker initialization is not performed explicitly. Instead it is integrated into the UPDATE method of the tracker, which is called in the first iteration of the main loop ($t = 0$). The reason for this is, that the initialization needs to be performed not only at $t = 0$, but also when new targets appear after all camera frames were empty for several subsequent time steps.

5.3.2.2 Tracker Prediction Step

After initialization, the tracker can predict target locations in a new set of frames F for $t > 0$ as shown in algorithm 3. In this prediction step, the number of targets K, the box IDs I, the box valid states V and the match counter of each core tracker stay unchanged as compared to the previous time step $t - 1$. Appearing and disappearing targets, which affect these quantities, are handled by the update mode of the

Algorithm 2: Tracker initialization ($t = 0$).

```
procedure INITIALIZE($F, \mathcal{D}'$)
    $N \leftarrow$ LENGTH_FIRST_DIMENSION($F$)
    $K \leftarrow K'$
    $B \leftarrow B'$
    $I \leftarrow [\,]_{N \times K}$
    $V \leftarrow [\,]_{N \times K}$
    core_trackers $\leftarrow [\,]_{N \times K}$
    match_count $\leftarrow [\,]_{N \times K}$
    for each $j \in \{1, 2, \ldots, N\}$ do
        for each $k \in \{1, 2, \ldots, K_j\}$ do
            $I_{j,k} \leftarrow k - 1$
            $V_{j,k} \leftarrow$ false
            core_trackers$_{j,k}$ $\leftarrow$ new core_tracker($\mathcal{A}_{\text{trk}}$)
            core_trackers$_{j,k}$.INIT($F_j, B_{j,k}$)
            match_count$_{j,k}$ $\leftarrow 1$
        end for
    end for
end procedure
```

tracker. The only purpose of the prediction step is to predict the new bounding box coordinates B of each target in the frames F by calling the PREDICT method of the assigned core tracker. This method is provided by the application programming interface (API) of the core tracker and not to be confused with the PREDICT method of the tracker itself, as used in algorithm 1.

Algorithm 3: Tracker prediction step ($t > 0$).

```
procedure PREDICT($F$)
    $B \leftarrow [\,]_{N \times K}$
    for each $j \in \{1, 2, \ldots, N\}$ do
        for each $k \in \{1, 2, \ldots, K_j\}$ do
            $B_{j,k} \leftarrow$ core_trackers$_{j,k}$.PREDICT($F_j$)
        end for
    end for
end procedure
```

5.3.2.3 Tracker Update with Detections

To prevent accumulation of large tracking errors, the core trackers are updated every f_d steps based on the bounding boxes B' returned by the object detector. Algorithm 4 shows this update procedure, which is largely inspired by [120]. As mentioned before, the update procedure implicitly handles initialization of the tracker, whenever none of the camera scenes contained a tracking target in the previous time step $t-1$. If there was at least one target in any of the camera scenes at $t-1$, the tracker is first brought into the next state via its PREDICT method to match the time t of the detection results. Then, the MATCH_BOXES method is called for each frame $F_j \in F$ and tries to associate each detected bounding box in B'_j with a bounding box in B_j of an existing tracking target via the Hungarian method [54]. Matching of bounding boxes is performed separately for each frame in F and no cross-optimization between individual frames is performed. Thus, three sets of indices $\mathcal{M}$, $\mathcal{M}_{\text{det}}$ and $\mathcal{M}_{\text{trk}}$ are returned for each frame. The first set contains pairs of indices (δ, τ), which indicate, that the bounding box $B_{j,\tau}$ of the τth tracking target is associated with the δth detected bounding box $B'_{j,\delta}$. The set $\mathcal{M}_{\text{det}}$ contains indices δ of detected bounding boxes in B'_j, which could not be matched with any bounding box B_j of the current tracking targets. This might be due to a false positive detection or because a new target appeared and is detected for the first time. Similarly, the set $\mathcal{M}_{\text{trk}}$ contains indices τ of tracked bounding boxes B_j, which could not be matched with any of the detected bounding boxes in B'_j. Reason for this can be a false negative detection or a target, which left the scene and is not detected anymore. Each of the three cases has to be handled separately during update of the tracker.

For every new detected target in $\mathcal{M}_{\text{det}}$, a new core tracker is created, initialized with the according detected bounding box $B'_{j,\delta}$ and appended to the list of all core trackers. Similarly, the detected bounding box $B'_{j,\delta}$ is appended to the list of tracked bounding boxes B_j and its validity state is set to false. The match counter of the core tracker is initialized to 1 and the lowest unused box ID is found and assigned to the new target. Finally, the number K_j of tracking targets for the jth frame in F is incremented by one.

In case of a match between a detected bounding box $B'_{j,\delta}$ and the bounding box $B_{j,\tau}$ of an already tracked target, a state update of the corresponding core tracker has to be performed. This is done by calling the INIT method of the core tracker with the according frame of F and the detection box $B'_{j,\delta}$. To prevent a core tracker from being updated with a false positive detection, a state update is only performed, if the target is valid. In case of an invalid target, instead the associated match counter is incremented by one every time this target is matched to any detected bounding box. When the match counter reaches the hypothesis valid threshold θ_v, the target becomes valid and an update of the assigned core tracker is performed. Any further redetection of this target leads to an immediate state update of the according core tracker without an increment of the match counter. This solution can be seen as a probation period of length θ_v, that each newly created target has to undergo, before it can affect the overall tracking result.

If a target is not detected in an update step, that is its index τ is present in $\mathcal{M}_{\text{trk}}$, its associated match counter is decreased by one. If more than θ_d subsequent misses of a particular target occur, it is deleted. Additionally, the core tracker, match counter, bounding box, box ID and box validity state associated with this target are deleted.

In general, the match counter of each core tracker acts as a low-pass filter, which is parameterized by the two hypothesis thresholds θ_v and θ_d. It prevents a state update of the corresponding core tracker with faulty detections, which occur only in a single or a few subsequent frames. Without this low-pass filter every false positive detection would result in the creation of a *ghost target*, which would exist in the frame during the f_d intermediate prediction steps and only disappear on the next update step. Likewise, a missed detection of an existing target during the update step would lead to the immediate deletion of the target. This is unwanted, because the target might still be visible in the scene, but was simply missed by the object detector. The price to pay for the prevention of these effects by the low-pass filter is a decreased reaction time for detecting newly appearing targets and deleting targets, which leave the scene. This is because the match counters have to be increased or decreased according to the values θ_v and θ_d before such an event can be registered. An example of a full run-in period of the filter is shown in fig. 5.5. First, all targets are invalid, indicated by orange bounding boxes. Most of them get redetected, when the detector is run in every second frame, and the associated match counters are incremented. When the counter values reach the hypothesis valid threshold $\theta_v = 3$, the targets change their states to valid, indicated by a green bounding box.

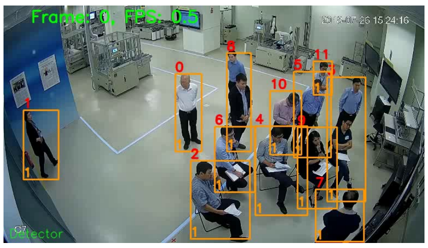
The mentioned MATCH_BOXES function associates bounding boxes B'_j of detected targets with bounding boxes B_j of existing tracking targets in each frame $F_j \in F$. It does so by first computing a cost matrix

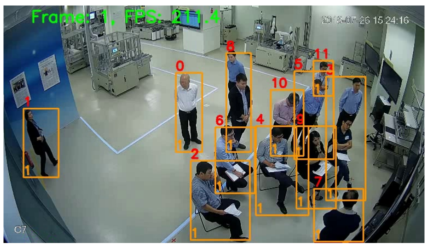
$$\widehat{C}_{j,\delta,\tau} = -\text{IoU}\left(B'_{j,\delta}, B_{j,\tau}\right) \tag{5.1}$$

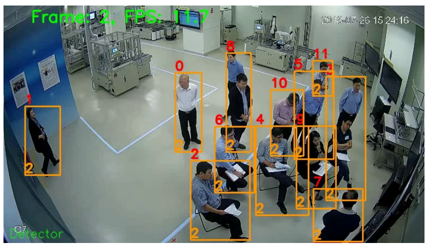
Algorithm 4: Tracker update with detections.

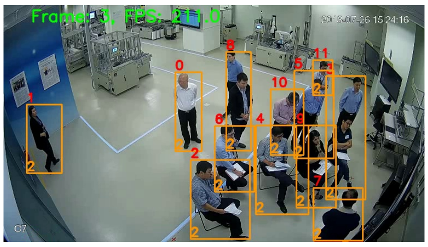
```
procedure UPDATE($F, \mathcal{D}'$)
    if $K_j == 0\ \forall\ j \in \{0, 1, \ldots, N\}$ then
        tracker.INITIALIZE($F, \mathcal{D}'$)                                  (sec. 5.3.2.1)
    else
        tracker.PREDICT($F$)                                                   (sec. 5.3.2.2)
        for each $j \in \{1, 2, \ldots, N\}$ do
                                                ▷ Match detections and tracker predictions
            $\mathcal{M}, \mathcal{M}_{\text{det}}, \mathcal{M}_{\text{trk}}$ = MATCH_BOXES($B_j, B'_j, \theta_{\text{IoU}}$)
                                                                   ▷ Handle matches $\mathcal{M}$
            for each $(\delta, \tau) \in \mathcal{M}$ do
                if match_count$_{j,\tau} < \theta_v$ then
                    match_count$_{j,\tau}$ ← match_count$_{j,\tau} + 1$
                else
                    core_trackers$_{j,\tau}$.INIT($F_j, B'_{j,\delta}$)
                    $V_{j,\tau}$ ← true
                end if
            end for
                                                    ▷ Handle new detections $\mathcal{M}_{\text{det}}$
            for each $\delta \in \mathcal{M}_{\text{det}}$ do
                $k \leftarrow K_j \leftarrow K_j + 1$
                core_trackers$_{j,k}$ ← new core_tracker($\mathcal{A}_{\text{trk}}$)
                core_trackers$_{j,k}$.INIT($F_j, B'_{j,\delta}$)
                match_count$_{j,k}$ ← 1
                $B_{j,k} \leftarrow B_{j,\delta}$
                $I_{j,k}$ ← GET_LOWEST_UNUSED_BOX_ID()
                $V_{j,k}$ ← false
            end for
                                      ▷ Handle undetected tracking targets $\mathcal{M}_{\text{trk}}$
            for each $\tau \in \mathcal{M}_{\text{trk}}$ do
                if match_count$_{j,\tau} > \theta_v - \theta_d$ then
                    match_count$_{j,\tau}$ ← match_count$_{j,\tau} - 1$
                else
                    $K_j \leftarrow K_j - 1$
                    DELETE core_trackers$_{j,\tau}$
                    DELETE match_count$_{j,\tau}$
                    DELETE $B_{j,\tau}$
                    DELETE $I_{j,\tau}$
                    DELETE $V_{j,\tau}$
                end if
            end for
        end for
    end if
end procedure
```

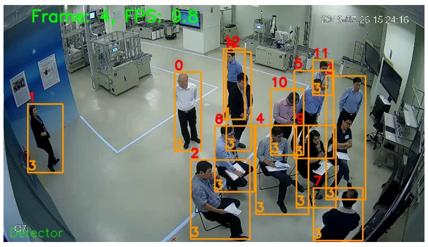
of the intersection over union (IoU) (see appendix A.3) of each possible match between the two partite sets of boxes in B'_j and B_j. This yields a bipartite graph with the boxes in B'_j and B_j as vertices of one of the partite sets each, and the cost matrix $\widehat{C}_j$ representing the weights of this graph. The negative sign

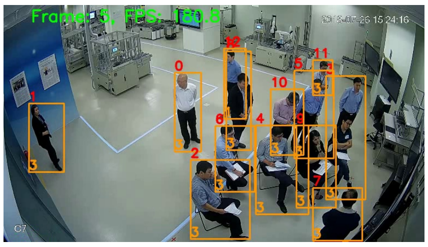

(a) $n = 0$, Update, $MC = 1$.

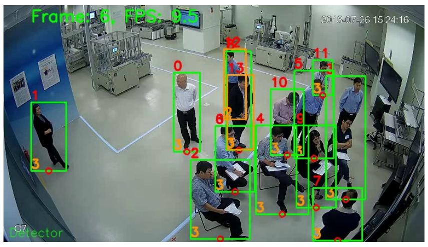

(b) $n = 1$, Predict, $MC = 1$.


(c) $n = 2$, Update, $MC = 2$.


(d) $n = 3$, Predict, $MC = 2$.


(e) $n = 4$, Update, $MC = 3$.


(f) $n = 5$, Predict, $MC = 3$.


(g) $n = 6$, Update, $MC = 3$.

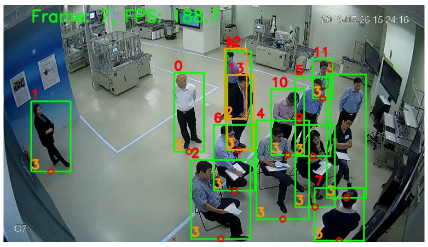

(h) $n = 7$, Predict, $MC = 3$.

Fig. 5.5: Example tracking output illustrating the low-pass filter in operation. Depicted are the first 8 frames in a video sequence acquired by camera C_7. In every second frame, the detector is run ($f_d = 2$) and the trackers are updated based on the detection results. In each intermediate frame, the tracker predicts the target location. On every successful redetection of a target, the match counter (MC) is incremented, until it reaches the hypothesis valid threshold $\theta_v = 3$. Then, the target states change from invalid (orange bounding boxes) to valid (green bounding boxes). Note, that the match counter usually differs for every target. However, in the initial frames (shown here), the match counters of most targets are equal. Thus, the captions state only one global counter value for each frame. That some of the targets were not redetected sucessfully, can be seen in frames 10 and 11, where two targets are still invalid.

indicates a high cost for low IoU of two boxes. Finding the optimal assignment of detected bounding boxes B'_j to boxes of existing targets B_j is equivalent to finding a complete assignment between the two partite sets of minimal cost, that is maxmimum total IoU between the two sets. The Python Scipy implementation of the Hungarian method [54, 121] is used to solve this classical linear sum assignment problem [122], yielding pairs of indices (δ, τ) indicating the matches between $B'_{j,\delta}$ and $B_{j,\tau}$. All pairs of boxes with an IoU greater or equal than the match IoU threshold θ_{IoU} are filtered out and form the match set $\mathcal{M}$. Indices of detected bounding boxes in B'_j, which could not be associated to any existing target box in B_j are aggregated in the set $\mathcal{M}_{\text{det}}$ of new detections. Similarly, indices of tracked boxes in B_j, which were not assigned to any detected bounding box in B'_j form the set $\mathcal{M}_{\text{trk}}$ of misses.

5.3.2.4 Retrieval of Tracking Results

The current status of the tracker can be retrieved by calling the methods GET_STATE_INFO and GET_-TARGETS after a prediction or update step. The first method simply returns the total number of tracking targets K, bounding boxes B, their IDs I and validity states V for each frame $F_j \in F$. The second method computes the world positions W of valid targets and a set of unique target IDs T as shown in algorithm 5. While the other variables differ in every frame of F, W and T are unified across all frames. The world position of a tracked person is represented by the location of the feet on the ground plane, which is estimated by computing the lower middle point (u, ν) of the person's bounding box. Retrieving the world position is now as easy as mapping (u, ν) onto the undistorted image coordinates (u', ν'), by means of the intrinsic camera parameters in P_j, and then perspectively transforming them onto the ground plane via the homography matrix $H_j \in P_j$. Section 5.7.5 gives a more detailed explanation of this mapping procedure. Next, invalid targets ($V_{j,k} =$ false) are filtered out, and a unique ID for the kth target in the jth frame of F is computed by means of Cantor's pairing function [123]. This function takes the frame index j and the box id $I_{j,k}$ of each target as an input and returns a target ID, which is unique across all frames in F. Advantage of this encoding is, that the target ID carries the information about the original frame index j and box ID $I_{j,k}$ of a target, which can later easily be retrieved by computing the inverse of the Cantor pairing function. Finally, world positions in W are merged across the frames in F, yielding a single set of world positions for all targets. The current implementation stacks the individual world positions in $W_j \in W$ vertically without handling multiple detections of the same target. This means, that a target, which is visible in multiple camera FOVs simultaneously, occurs multiple times in W and T.

Algorithm 5: Retrieval of tracking results.

```
procedure GET_TARGETS
    $W \leftarrow [\,]_{N \times K}$
    for each $j \in \{1, 2, \ldots, N\}$ do
        for each $k \in \{1, 2, \ldots, K_j\}$ do
                                        ▷ Compute foot point of each box as target location
            $(u_{\min}, \nu_{\min}, w, h) \leftarrow B_{j,k}$
            $(u, \nu) \leftarrow (u_{\min} + 1/2w, \nu_{\min} + h)$
                                        ▷ Transform points to world coordinates
            $W_{j,k} \leftarrow$ IMAGE_TO_WORLD$((u, \nu), P_j)$
        end for
    end for
                                        ▷ Filter out invalid targets and compute unique IDs
    $\widehat{k} \leftarrow 1$
    for each $j \in \{1, 2, \ldots, N\}$ do
        for each $k \in \{1, 2, \ldots, K_j\}$ do
            if $V_{j,k} ==$ true then
                $W_{j,\widehat{k}} \leftarrow W_{j,k}$
                $T_{j,\widehat{k}} \leftarrow$ CANTOR_PAIRING$(j, I_{j,k})$
                $\widehat{k} \leftarrow \widehat{k} + 1$
            end if
        end for
    end for

    $W \leftarrow$ MERGE_MULTIPLE_VIEWS$(W)$
    return $T, W$
end procedure
```

5.4 Database Layout

As mentioned in section 5.1, a MySQL database is used to permanently store tracking results. This database, called *sfmt*, uses the default InnoDB engine and contains a single table, named *trajectories*, with the column layout shown in table 5.3. Thus, each row in the database contains the world position $W_k \in W$ and target ID $T_k \in T$ of a single valid tracking target at time t and step number n, respectively. An additional column stores an auto incremented integer value as primary key of each database entry. As an example, consider a Monitoring system with 2 cameras and currently 3 and 4 valid targets in the first and second camera view. This results in 7 database entries per tracking step, and, hence, 7 entries with the same timestamp t and step number n, but unique primary keys. It is not necessary to store the frame index j and box ID $I_{j,k}$ of each valid target because they are encoded in the target ID T_k as explained in the previous section. Similarly, the number of valid targets $\widehat{K}$ can easily be deduced from the target IDs of rows with the same timestamp or step number. The number N of cameras is not stored either, because it is assumed constant for the entire lifespan of the monitoring system. Inferring N from the target IDs T is possible, but only when there is at least one valid target in every camera view. Furthermore, bounding boxes B are omitted in the database, because they consume a large amount of storage space and are not needed within the web apps.

To manage database access, a wrapper class based on the official MySQL Python connector [124] is implemented. This class has methods for the creation of the *sfmt* database and the *trajectories* table, as well as for the insertion of a new tracking data package $\mathcal{D}$. An instance of this wrapper class is used in the output thread of the main process to establish a connection to the database and insert every new tracking data package $\mathcal{D}$ (see section 5.2).

Tab. 5.3: Layout of the trajectories table in the MySQL database of the Shopfloor Monitor. A row of this table corresponds to the kth valid tracking target in the jth camera view.

Variable	Symbol	Column Name	Data Type	Not Null	Auto Increment	Primary Key
Primary Key	–	primary_key	Bigint(20) Unsigned	✓	✓	✓
Timestamp	t	timestamp	Datetime(6)	✓	✗	✗
Step Number	n	step_number	Bigint(20) Unsigned	✓	✗	✗
Target ID	T_k	target_id	Int(10) Unsigned	✓	✗	✗
World Position	X_k	x_pos	Double	✓	✗	✗
World Position	Y_k	y_pos	Double	✓	✗	✗

5.5 Web Applications

Web apps were introduced as front ends for the Shopfloor Monitor in section 5.1. They are served by a Python Flask app and communicate with the main process on the analysis PC via websockets and the MySQL database. Every client device with a modern web browser can access the web apps over the Internet, enabling the user to view and interact with the results of the detection and tracking system. Despite providing only a single visualization app at this stage of the project, the Shopfloor Monitor shall provide a variety of web apps in future. Each app is specifically geared to a certain task, or the requirements of a potential customer. Conceivable use cases of the Shopfloor Monitor and according web apps are explored in chapter 2. Each customer might purchase multiple web apps, which are bundled in a single admin web page. Scope of this section is to explain the implementation of this central admin page in detail, and this way, support the development of other web apps. Furthermore, the already implemented visualization app is presented.

5.5.1 App Framework

The admin page is a simple website based on HTML, CSS and Javascript, with the directory structure shown in the following tree.

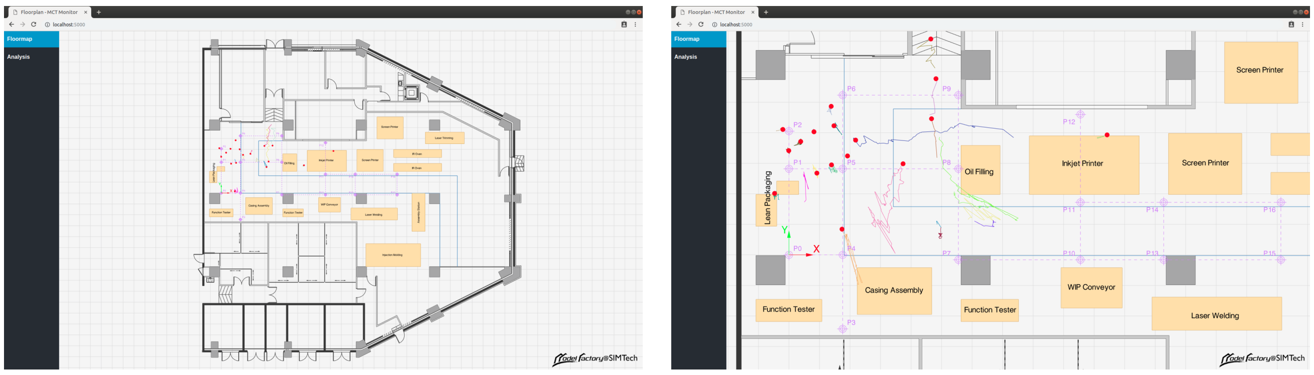
```
webapp
├── flask_server.py
├── templates
│   ├── index.html
│   └── floormap.html
└── static
    ├── logo.png
    ├── style.css
    └── floorplan
        ├── three.js
        ├── d3.min.js
        ├── detector.js
        ├── socket.io.min.js
        ├── floormap.js
        └── floorplan.json
```

This directory is located in the base directory of the Shopfloor Monitor. The central element is the `flask_server.py` file, which implements the Python Flask app for serving the admin page. In this script, first a Flask-SocketIO instance is instantiated, which connects to the Redis message broker and enables a socket connection between the client app and the main process of the Shopfloor Monitor. Task of the Flask app is to serve the HTML files of the admin page, which are located in the `templates` subdirectory. It uses an Eventlet webserver for doing so. This server can easily be configured to serve the app to any client device over the Internet. However, this introduces new challenges to keep the system secure. Thus, during development, apps are only served to localhost. Since only one template can be served at a time, a specific Uniform Resource Locator (URL) is used to select a template. Each URL is routed to another function within the Flask app, which serves one of the available templates. These functions also allow to dynamically pass data from the Flask app to the client. Individual web apps are pages within the admin website. That means, each HTML file within the `templates` subdirectory implements one web app. Because each web app needs to display the same sidebar and the same footer, these elements are described in a central `index.hmtl` file. The template inheritance feature of Flask's Jinja template engine allows to extend this central template by an app-specific template file, which implements its own page body, while using sidebar and footer from the `index.hmtl` file. Currently, only one such app-specific template, `floormap.hmtl`, file is used for the visualization app. A single style sheet is used to define element styles for the visualization app and the central elements, such as header, footer and logo. It is located in the `static` subdirectory. If more web apps are developed, each app should use an own style sheet to keep the website modular. It can then be loaded in in the header of the HTML template file of the web app. Apart from the style sheet, the `static` subdirectory contains more static files, which are served by the Flask app, and which can be used within the template files of apps. For example, a Portable Network Graphics (PNG) file of the logo is loaded in the `index.hmtl` file and displayed in the footer of every page. Furthermore, the entire `floorplan` subdirectory, containing Javascript files needed by the visualization app is statically served by the Flask app. These files are included in the header of the `floormap.hmtl` template file.

Adding a new web app to the Shopfloor Monitor is as easy as creating a new HTML template file in the `templates` subdirectory, routing a new url to this template by adding an according function to the `flask_server.py`, and placing a style sheet and necessary static files in the `static` subdirectory. Output data of the detection and tracking system can easily be accessed either via the MySQL database or via websockets. To use websockets, the web app needs to load the SocketIO Javascript library, located in `socket.io.min.js`. The socket connection can not only be used to transmit tracking data, but also to send other data between the client and the Flask server process. For example, a socket event is used to notify the visualization app about start and termination of the main process.

5.5.2 Visualization App

Currently, only one web app exists. It visualizes world positions of tracking targets in real-time on a rendered 2D floor plan of the factory. Figure 5.6 shows two screenshots of the visualization app at different zoom levels. The current world position of each target is represented by a red circle, and the past 200 locations are visualized as a colored trace. These colors are allocated randomly, but stay constant as long as the target ID stays the same. Mouse events, such as zoom and pan, allow the user to interact with the floorplan and observe regions of interest in more detail. The reason for developing such a visualization app first, is the illustrative presentation of the detection and tracking system, which aids the development process.



(a) Zoom level showing the entire floor plan. **(b)** Zoomed into region of interest.

Fig. 5.6: Screenshots of the visualization web app running in Google Chrome. This app displays target trajectories in real-time on a 2D floor plan and allows user interactions, such as zoom and pan.

To render the model of the factory floorplan within the client browser, the visualization app uses the Javascript Library Three.js [125]. This library is a wrapper around WebGL [126], a cross-plattform, low-level 3D graphics API, and provides convenient functions for the creation and rendering of arbitrary 2D and 3D scenes. Three.js is located in the `three.js` file in the `static` subdirectory along with the main javascript file `floormap.js`. This main script implements the visualization of the floorplan as defined by the Javascript Object Notation (JSON) file `floorplan.json`, as well as the dynamic plotting of target locations. The SocketIO Javascript library is used to receive data from the Flask app via websockets. Mouse events, such as pan and zoom, are captured with the help of the Data-Driven Documents Javascript library (D3.js) [127] and handled within the main Javascript. The Detector Javascript in `Detector.js` is provided by Three.js and checks for availability of WebGL on the client device. All Javascript files are loaded within the header of the template file `floormap.html`. Here, a single HTML canvas element is created for the display of the rendered floorplan.

To draw an element, such as a line, within a Three.js model, first its geometry and a material have to be specified. From this, the element instance can be created and added to the scene. In addition, the scene is equipped with lights illuminating the model, and a camera capturing a view of the model. When all elements have been added to the scene, a single call to the rendering engine renders a view of the model and outputs it to the HTML canvas. To allow for flexible floorplan models, the model is not hard-coded within the `floormap.js`, but instead it is built dynamically from element definitions in the model file, called `floorplan.json`. Here, the whole model is described as a list of dictionaries, which contain the properties of one element each. Possible elements are lines, polygons, arcs, markers, arrows and text. In this work, the model file is created manually from an AutoCAD drawing due to the absence of a conversion plugin. Within the `floormap.js`, a loop iterates over all dictionaries in the model file and creates according geometries, materials and elements, which are then added to the scene. To create a 2D model, z coordinates of all elements are set to 0 and an orthographic camera, which does not introduce any perspective distortion, is used. By updating the parameters of the camera viewport, the model view can be zoomed and panned.

Besides creating and rendering the floorplan model, the `floormap.js` Javascript draws and updates position markers and trajectories of the tracking targets, whenever a new data package $\mathcal{D}$ is received from the main process via the websocket. This procedure is shown in algorithm 6. On receive of a new data package $\mathcal{D}$, containing target IDs T and world positions W of the currently tracked targets, the RECEIVED__DATA__PACKAGE event is called. Here, the DIFF function compares the received target IDs with those of the last time step and classifies them into added, updated and removed IDs. For each target ID in the list of added IDs, a new trajectory element is created and added to the scene. Such a trajectory consists of a randomly colored line and a red circle at the current world position of the target. Each target in the list of updated IDs already has an according trajectory element, which needs to be updated. Thus, a new point at the current target location is added to the trajectory line and the red position marker is shifted to this point. If the trajectory line exceeds a length of 200 points, the oldest point is removed from the line. If a target ID is removed, its associated trajectory is not deleted immediately, but instead decayed slowly. As long as the line contains points, the oldest point is removed, whenever the DECAY__OLD__TRAJECTORIES method is called. When the main process terminates, it triggers the CLOSED socket event, which is used to remove all trajectories and position markers from the scene.

5.6 Camera Setup

The Shopfloor Monitor utilizes two pan-tilt-zoom (PTZ) IP cameras of type Dahua DH-SD22204T-GN, which are referred to as C_3 and C_7, relating to the channel numbers of the cameras in a larger camera network. Both cameras have a resolution of 1920×1080 pixels. They are set to a fixed frame rate of 15 Hz and H.264H video codec with a variable bit rate of 640 kbit/s to 8192 kbit/s is used. Scalable Video Coding (SVC) is disabled and image quality is fixed to the highest possible setting. The Intra Frame (IFrame) Interval is set to a value of 50, which means, that only every 50th frame is transmitted entirely without reference to other frames. All 49 subsequent frames are described by their differences from the previous Intra Frame to reduce the amount of transferred data. Substreams are disabled to reduce the produced amount of data further. This enables transmission of the main stream at a high bit rate and low latency. Further details about the optical parameters, network protocols, power ratings, PTZ functions and special functions, such as night vision, automatic white balance, gain control and noise reduction, are listed in the datasheet in appendix C.3. Both cameras allow dynamic setup of their fields of views (FOV) by changing the camera orientation and focal length (zoom). The FOVs are manually set up to cover as most of the ground plane of the factory as possible. Both FOV are slightly overlapping. The pan, tilt and zoom settings are stored in a profile and can thus be easily recovered after a system shutdown, which makes it possible to reuse the same calibration parameters without the need for recalibration. Since both cameras provide a network interface based on the real-time streaming protocol (RTSP) [128], their video streams can be directly accessed with a compatible software module, skipping the installed network video recorder (NVR) for the sake of a lower latency and a higher transmission bandwidth.

Algorithm 6: Dynamic drawing of target trajectories.

```
Data: socket
  id_pool ← [ ]
  procedure socket.RECEIVED_DATA_PACKAGE
    (added, updated, removed) ← DIFF(id_pool, $T$)
    for each added_id in added do
      id_pool.PUT(added_id)
      CREATE_TRAJECTORY(added_id, $W_{\text{added\_id}}$)
    end for
    for each updated_id in updated do
      UPDATE_TRAJECTORY(updated_id, $W_{\text{updated\_id}}$)
    end for
    for each removed_id in removed do
      REMOVE_TRAJECTORY(removed_id)
      DELETE id_pool[removed_id]
    end for
    DECAY_OLD_TRAJECTORIES()
    RENDER()
  end procedure

  procedure socket.CLOSED
    id_pool ← [ ]
    REMOVE_ALL_ELEMENTS_FROM_SCENE()
    RENDER()
  end procedure
```

5.7 Camera Calibration

Camera calibration is performed to compute the set of parameters P needed to map target locations from image coordinates to the factory ground plane. The parameters P are part of the non-linear algebraic equation system, which models the image formation process in each camera. Parameters are estimated iteratively based on corresponding points in the images and on the factory ground plane. All functions descibed below are implemented in the OpenCV image processing library [49].

Section 5.7.1 introduces the camera model. A simplification by means of using a homography is presented in section 5.7.2. Intrinsic camera parameters are estimated in section 5.7.3 and the ground plane homographies are computed in section 5.7.4. Section 5.7.5 explains the mapping procedure of target locations form image to world coordinates. The calibration results are validated in section 5.7.6, and finally, section 5.7.7 discusses the optimal calibration interval.

5.7.1 Camera Model

The camera model describes the geometric relations between the factory ground plane and the camera image. Having a horizontal angle of view of 93.5° (see appendix C.3), the cameras used in the Shopfloor Monitor introduce a significant image distortion. Thus, a fisheye camera model is used [129]. This model is implemented in the OpenCV fisheye calibration package [130], which is used throughout the calibration procedure.

The camera model uses the three different coordinate systems shown in fig. 5.7. First, the world coordinate system, which describes the actual position (X, Y, Z) of an object in the observed scene. Second, the camera coordinate system, which is fixed with respect to the camera body and describes the position (x, y, z) of the same object with respect to the camera. And third, the image coordinate system, in which the object occurs at the pixel positions (u, ν).

To map a point (X, Y, Z) from the world coordinate system to the same point (x, y, z) in the camera coordinate system, a rotation and a translation are necessary as defined by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathcal{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathcal{T}. \tag{5.2}$$

Here $\mathcal{T} \in \mathbb{R}^3$ is a translation vector and $\mathcal{R} \in \mathbb{R}^{3\times3}$ a rotation matrix. These are the so called *extrinsic* parameters of the camera.

To map the same point from the camera into the image coordinate system two effects have to be considered. First is the characteristics of the projection of the light rays from the scene onto the image sensor, which is modeled via a *camera matrix* $M \in \mathbb{R}^{3\times4}$ and second is the radial and tangetial distortion of the image due to the specific lens geometry. Especially in a wide angle camera as the one used here, image distortion plays a significant role.

First, a pinhole camera model is assumed to project the point (x, y, z) from the camera coordinate system onto the image plane, yielding the point (x', y') with respect to the camera coordinate system. The projection is done by normalizing with the z-coordinate according to

$$x' = \frac{x}{z},\ y' = \frac{y}{z}. \tag{5.3}$$

To map the point (x', y') from camera coordinates to image coordinates (u, ν), first a fisheye distortion has to be applied, yielding the new point (x'', y'') in camera coordinates with

$$x'' = \frac{\Theta_d}{r}\, x',\ y'' = \frac{\Theta_d}{r}\, y'. \tag{5.4}$$

The fisheye distortion is modeled as radial distortion, which means every pixel is moved outwards perpendicular to a circle around the image center. The further away a pixel from the center is, the further it is moved outwards. Therfore, the distortion coefficient Θ_d can be computed by the following definitions

$$r^2 = x'^2 + y'^2, \tag{5.5}$$

$$\Theta = \arctan(r), \tag{5.6}$$

$$\Theta_d = \Theta + k_1\Theta^3 + k_2\Theta^5 + k_3\Theta^7 + k_4\Theta^9. \tag{5.7}$$

Here, k_1, k_2, k_3 and k_4 denote the four distortion coefficients. After application of the fisheye distortion, the point (x'', y'') is still defined with respect to the camera coordinates system. Hence, a further transformation has to be computed to get the image coordinates (u, ν) of this point. Such a transformation is defined as follows

$$u = f_x\,(x' + \alpha y') + c_x,\ \nu = f_y y' + c_y, \tag{5.8}$$

whereby f_x and f_y are the focal length in x, respectively y direction of the camera coordinate system, c_x and c_y define the principle point of the image and α is the skew coefficient. The first four parameters are combined in the camera matrix M.

The distortion parameters k_1, k_2, k_3, k_4, the skew coefficient α and the parameters f_x, f_y, c_x, c_y of the camera matrix M are known as the *intrinsic* parameters of the camera.

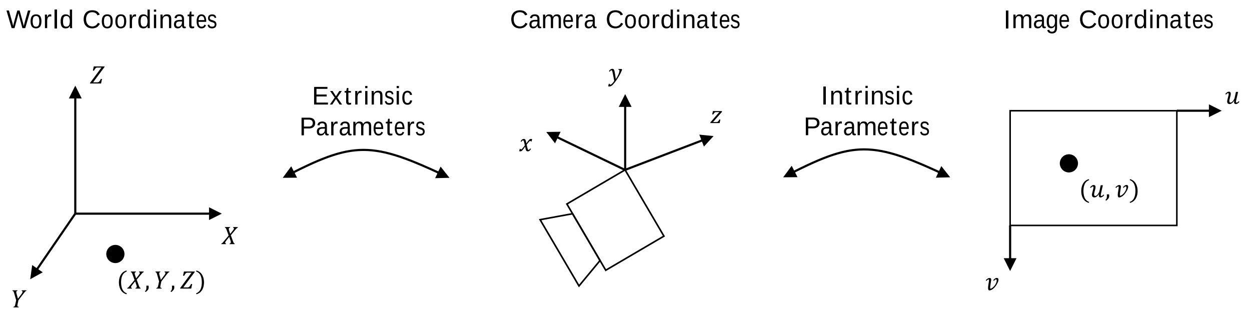


Fig. 5.7: Overview of the three coordinate systems of the fisheye camera model. A point (X, Y, Z) is mapped from world coordinates to the camera coordinates (x, y, z) as described by the extrinsic model parameters and then mapped to the image coordinates (u, ν) via distortion and projection as described by the intrinsic model parameters.

5.7.2 Ground Plane Homography

Many existing people-tracking systems assume a planar world constraint [131–134]. This means, i) the ground is planar and ii) people always have contact to the ground ($Z = 0$). The constraint is needed, as it is not possible to directly infer a target's 3D world position from a single 2D image without making any prior assumption about one of the world coordinates. This constraint also enables to model the mapping from image coordinates to the factory ground plane via a homography matrix $H \in \mathbb{R}^{3\times3}$. However, this requires undistorted image coordinates (u', ν'), which can be computed from the original image coordinates (u, ν) by reverting the effect of lens distortion. The homography can then be written as

$$s \begin{bmatrix} u' \\ \nu' \\ 1 \end{bmatrix} = H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{5.9}$$

Note, that the points are converted from Cartesian to homogenous coordinates, hence a 1 is appended. The parameter s is a scaling factor. If the image size is changed, all parameters of the homography matrix H have to be scaled accordingly.

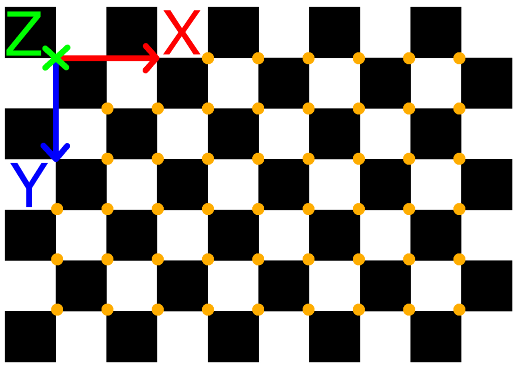
In the following calibration process, first the intrinsic camera parameters of each camera in the Shopfloor Monitor are determined to compute the undistorted image coordinates (u', ν') from the original, distorted image coordinates (u, ν). Afterwards, the homography matrices for both cameras are estimated.

5.7.3 Estimating Intrinsic Parameters

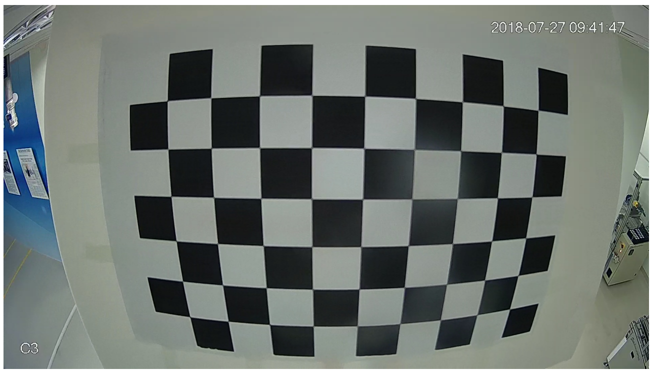
Estimation of the intrinsic camera parameters k_1, k_2, k_3, k_4, f_x, f_y, c_x, c_y and α requires a set of corresponding points on a planar surface in world and image coordinates. These correspondences are inserted into the camera equations. The unknown intrinsic parameters are obtained from these equations by iteratively minimizing the reprojection error between observed world points and the corresponding projected points with the Levenberg-Marquardt algorithm [129, 135, 136].

Corresponding points are acquired with a planar 9×6 chessboard pattern, which is printed out in A3-format and applied to a rigid cardboard to ensure planarity of the surface. The board is placed in different distances and angles front of the cameras C_3 and C_7, and 87 respectively 65 calibration images are taken. Figure 5.8 shows an example calibration image and the chessboard pattern.

The image points of the chessboard pattern are detected automatically with the OpenCV findChessboardCorners function and corresponding world coordinates are known with respect to the pattern



(a) 9 × 6 chessboard pattern used for intrinsic calibration.



(b) Example of an acquired calibration image.

Fig. 5.8: The chessboard pattern used to calibrate the intrinsic camera parameters and an example image of this pattern taken by camera C_3. The arrows in the chessboard indicate the pattern coordinate system and the orange circles mark the calibration points.

coordinate system. Position and orientation of the pattern coordinate system with respect to the factory ground plane are not needed for intrinsic calibration. The same holds for the exact scaling of the pattern. Thus, unit length of the pattern squares is assumed. In total 4698 and 3510 point correspondences are found for camera C_3 and C_7, respectively. They are fed into the CALIBRATE method of the OpenCV fisheye module, which returns an estimate of the intrinsic camera parameters (see table 5.4).

Tab. 5.4: Overview of the estimated intrinsic camera parameters for both cameras C_3 and C_7.

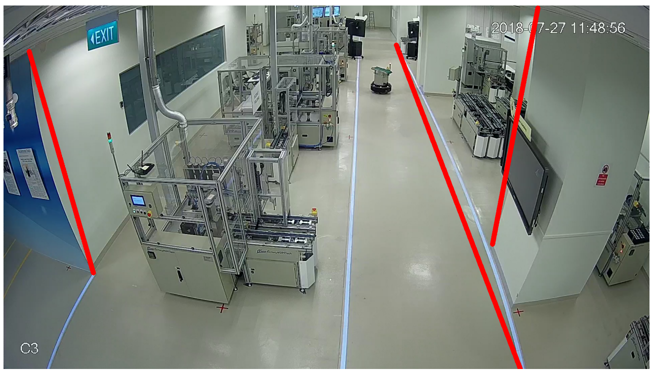
Camera	f_x	f_y	c_x	c_y	k_1	k_2	k_3	k_4	α
C_3	1103.13	1103.25	995.160	510.188	−0.116781	0.00397668	−0.00304310	0.00159480	0
C_7	1078.68	1078.64	982.458	450.170	−0.118950	0.00825041	−0.00245857	−0.00058231	0

The so computed intrinsic parameters can now be used to revert the distortion induced by the wide-angle lens. For this, an inverse mapping is computed with the INITUNDISTORTRECTIFYMAP function of the fisheye calibration package. It returns maps, which can be used by the REMAP function to undistort an image as shown in fig. 5.9. Undistortion leads to black areas at the image boundaries. The size of these areas can be controlled by the *balance* parameter $\alpha_B \in [0, 1]$ in the ESTIMATENEWCAMERAMATRIX-FORUNDISTORTRECTIFY function, which is used to determine a new camera matrix $M' \in \mathbb{R}^{3\times4}$ prior to computing the undistortion maps. Setting the balance parameter to 0, crops the output image, so that there are no black regions at the image boundaries. Setting it to 1 keeps all pixels from the original distorted image and large black regions occur.

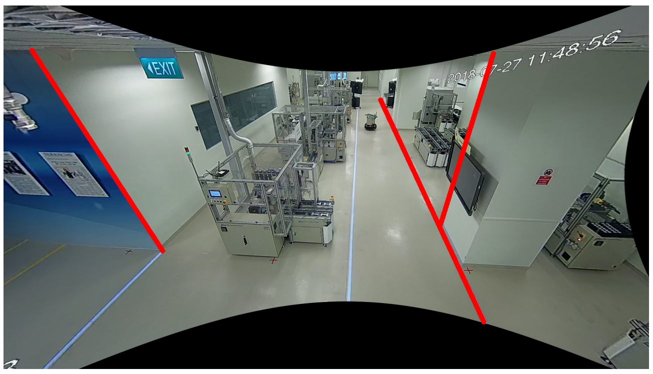
If only a sparse set of points rather than an entire image needs to be undistorted, the function UNDISTORTPOINTS of the OpenCV fisheye module can be used with the according intrinsic parameters as arguments.

5.7.4 Estimating Ground Plane Homographies

Estimation of the ground plane homography H for each camera requires a set of points in undistorted image coordinates (u', v') and the corresponding world positions (X, Y) on the ground plane. Instead of using a chessboard pattern, 17 calibration markers made from red tape are placed on the ground floor as shown in fig. 5.10. The corresponding locations on the ground plane are depicted in fig. 5.11. This plan also shows the world coordinate system, which originates at calibration point P_0.

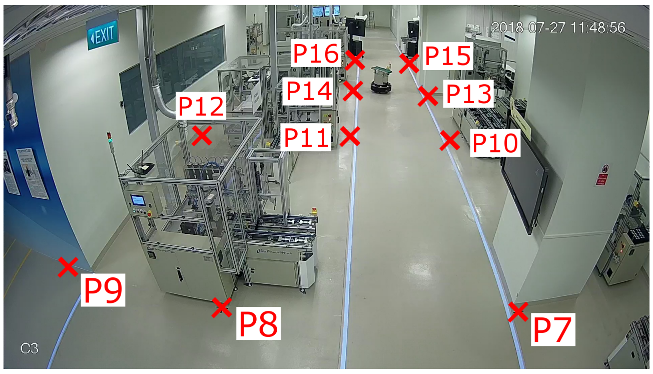


(a) Original distorted Camera Image.

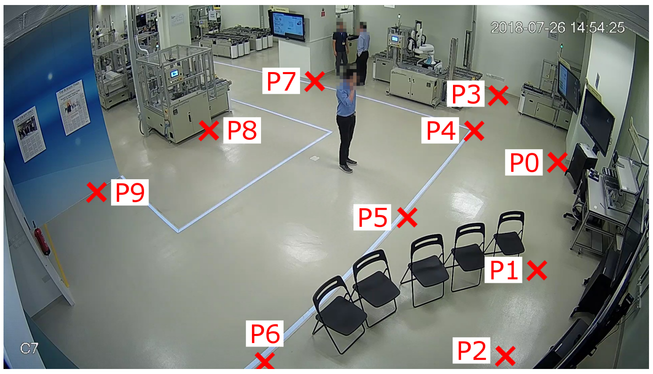


(b) Camera Image after Undistortion.

Fig. 5.9: Comparison of the original distorted camera image (a) and the undistorted camera image (b). Lines, which are straight in reality occur curved in the distorted image, whereby the curvature is greater at the image boundaries. In the undistorted image those lines occur straight again as can be seen by means of the red reference lines.



(a) Extrinsic calibration targets for camera C_3.



(b) Extrinsic calibration targets for camera C_7.

Fig. 5.10: Markers for extrinsic calibration of both cameras on the ground plane. The marker positions in these two images are used for the extrinsic calibration of both cameras.

To acquire calibration points a single image per camera is required and image coordinates (u, ν) of each marker are read out with the help of an image editing program. These points are undistorted, yielding the undistorted image points (u', ν'). Table 5.5 shows the positions of all calibration markers in world coordinates (X, Y), image coordinates (u, ν) and undistorted image coordinates (u', ν'). The marker points P_7, P_8 and P_9 occur twice, because they are visible in both camera FOVs. Due to being measured manually, the marker world positions have large errors of up to ± 50 mm. The measurements in image coordinates are more accurate with an error of ± 1 pixel.

The homographies H_3 and H_7 for both cameras are computed from the corresponding world coordinates (X, Y) and undistorted image coordinates (u', ν') by means of the findHomography function in OpenCV. This function uses the least median of squares (LMeDS) algorithm [137] to find an initial estimate for the homographies, which is further optimized by the Levenberg-Marquardt algorithm to minimize the reprojection error. The resulting homography matrices for cameras C_3 and C_7 are

$$H_3 = \begin{bmatrix} -1.58717 & -16.1157 & -16042.8 \\ 28.062 & -12.7544 & -27879.6 \\ 3.62252 \times 10^{-5} & -0.00698769 & 1 \end{bmatrix}, \quad H_7 = \begin{bmatrix} 30.6712 & 4.78286 & -46547.6 \\ 23.1779 & -85.44 & 6836.43 \\ 5.06243 \times 10^{-5} & -0.00953415 & 1 \end{bmatrix}.$$

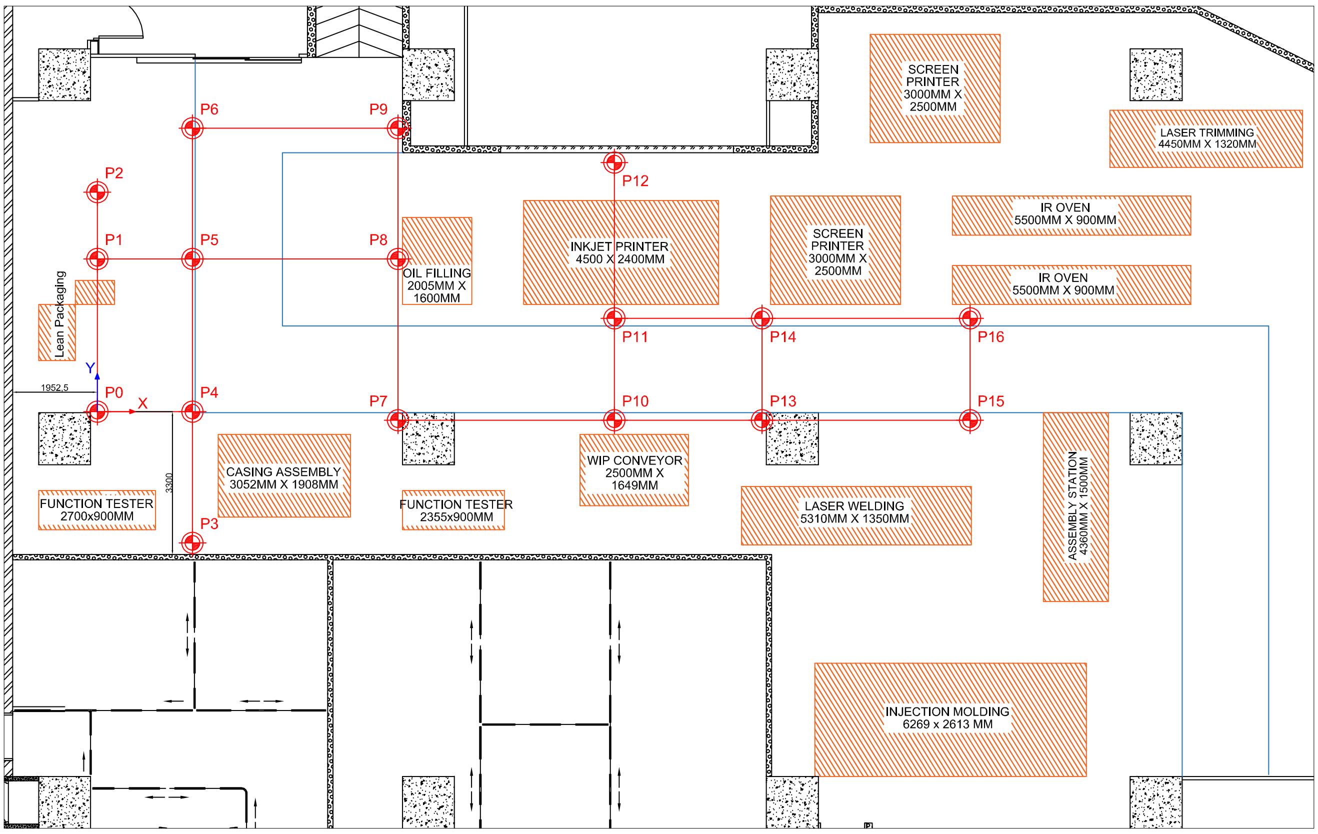

SCREEN PRINTER 3000MM X 2500MM
LASER TRIMMING 4450MM X 1320MM
P6
P9
P2
P12
IR OVEN 5500MM X 900MM
INKJET PRINTER 4500 X 2400MM
SCREEN PRINTER 3000MM X 2500MM
P1
P5
P8
OIL FILLING 2005MM X 1600MM
IR OVEN 5500MM X 900MM
Lean Packaging
P11
P14
P16
Y
1952.5
P0
X
P4
P7
P10
P13
P15
CASING ASSEMBLY 3052MM X 1908MM
WIP CONVEYOR 2500MM X 1649MM
3300
FUNCTION TESTER 2700x900MM
FUNCTION TESTER 2355x900MM
LASER WELDING 5310MM X 1350MM
ASSEMBLY STATION 4360MM X 1500MM
P3
INJECTION MOLDING 6269 x 2613 MM

Fig. 5.11: Section of the floor plan of the factory with positions of extrinsic calibration markers (red markings), installed machines (orange shaded boxed) and the pedestrian way (blue lines).

5.7.5 Mapping of Image to World Coordinates

The estimated intrinsic camera parameters and the ground plane homographies can now be used to map an image point (u, ν) to the according point (X, Y) on the factory ground plane. This process is shown in fig. 5.12. First, the image point (u, ν) is undistorted by means of the UNDISTORTPOINTS function of the OpenCV fisheye module. This yields the undistorted image point (u', ν'). The UNDISTORTPOINTS function reverts the mulitplication of the point with the camera matrix M as well as the radial and tangential distortion introduced by the camera lens. The new camera matrix M' is used to project the point into the undistorted image coordinates based on the *balance* parameter. The undistorted point (u', ν') is then projected onto the ground plane position (X, Y) by means of the ground plane homography eq. (5.9), which is realized with the OpenCV function PERSPECTIVETRANSFORM.

5.7.6 Validation of Calibration Results

The calibration results need to be evaluated to ensure a sufficient accuracy when mapping target locations from camera images onto the factory ground plane within the visualization app. First, intrinsic parameters are evaluated, then the ground plane homography, and finally, accuracy is computed on an independent set of validation points.

Tab. 5.5: Positions of the calibration markers for extrinsic calibration of both cameras on the ground plane, in the distorted image and in the undistorted image.

Camera	Marker	Ground Plane		Distorted Image		Undistorted Image	
		X/mm	Y/mm	u/pix.	ν/pix.	u'/pix.	ν'/pix.
C_7	P_0	0	0	1649	467	1443.970	471.755
	P_1	0	3525	1587	788	1410.840	690.134
	P_2	0	5065	1493	1042	1376.740	897.275
	P_3	2200	−3030	1473	271	1298.850	351.387
	P_4	2200	0	1400	376	1243.700	417.255
	P_5	2200	3525	1200	631	1121.660	561.677
	P_6	2200	6540	776	1068	864.012	867.471
	P_7	6940	−200	924	228	967.661	337.666
	P_8	6940	3525	611	375	786.602	417.362
	P_9	6940	6540	278	555	513.349	532.962
C_3	P_7	6940	−200	1531	912	1383.720	787.912
	P_8	6940	3525	651	899	802.927	756.779
	P_9	6940	6540	193	779	373.200	729.825
	P_{10}	11940	−200	1329	406	1214.040	452.562
	P_{11}	11940	2150	1028	392	1036.560	447.173
	P_{12}	11940	5750	591	386	775.236	439.106
	P_{13}	15350	−200	1259	269	1173.290	371.827
	P_{14}	15350	2150	1038	257	1042.660	368.371
	P_{15}	20150	−200	1206	171	1143.730	311.562
	P_{16}	20150	2150	1046	163	1047.920	309.851

5.7.6.1 Validation of Intrinsic Calibration

To examine the accuracy of the intrinsic calibration, the average intrinsic reprojection error for each calibration image can be calculated as follows

$$e_k = \frac{1}{n}\sum_{j=0}^{n-1} ||p_j - q_j||. \tag{5.10}$$

Here, $n = 54$ denotes the number of calibration points per images, $p \in \mathbb{R}^2$ is the measured position of a calibration point in image coordinates and $q \in \mathbb{R}^2$ is the corresponding reprojected point in image coordinates. Reprojection of a point means, it is first projected from image to world coordinates and then projected back from world to image coordinates by means of the estimated intrinsic and extrinsic parameters for the given image. As the model parameters do not exactly represent the actual physical parameters of the camera, an error remains. This is the reprojection error. If the model is very accurate, the reprojected point q lies close to its measured position p, so the reprojection error is low. However, if the model is inaccurate, the projected point q is far away from its measured position p and the error is large.

Figure 5.13a shows a plot of the intrinsic reprojection error for each of the 87 calibration images for camera C_3 and fig. 5.13b the according plot of the 65 calibration images for camera C_3. Additionally, the mean reprojection errors for all images are shown as horizontal red dashed lines.

The average intrinsic reprojection errors for camera C_3 and C_7 are 2.561 pixel and 2.056 pixel, respectively. The slightly larger error for camera C_3 might be caused by the fact that the calibration images are slightly

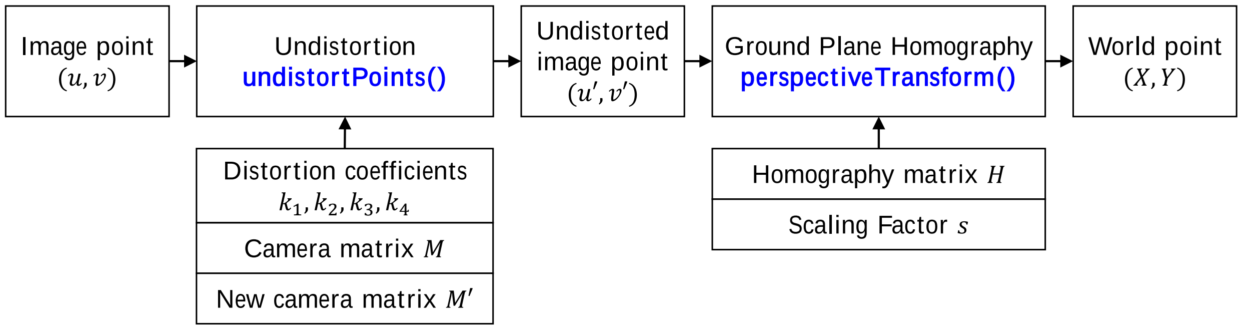
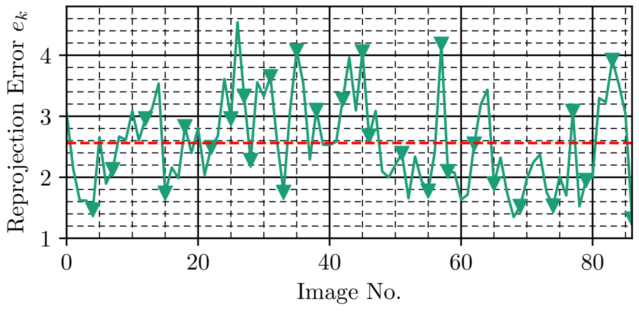
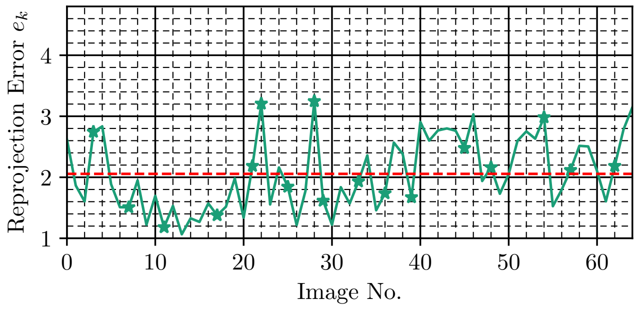


Fig. 5.12: Transformation of a point from image to world coordinates.



(a) Intrinsic reprojection error of camera C_3.



(b) Intrinsic reprojection error of camera C_7.

Fig. 5.13: Average intrinsic reprojection errors of intrinsic camera calibration per image. The horizontal red lines show the reprojection error averaged over all images.

blurry as the chessboard pattern was fixed to a long pole when taking the calibration images. However, the errors are in both cases low enough to confirm sufficient accuracy of the intrinsic calibration.

5.7.6.2 Validation of the Ground Plane Homography

Accuracy of the ground plane homographies H can be determined by mapping the image positions (u, ν) of the calibration markers onto the ground plane (X, Y) according to the method described earlier. Then, the euclidean distance between the actual position of the marker on the ground plane and the projected position can be computed and used as a measure for the error. Table 5.6 shows the actual world positions, the projected world positions and the error for the calibration markers. The average errors for C_3 and C_7 are 13.158 mm and 26.085 mm, respectively. They contain the intrinsic errors, as the intrinsic parameters are used to undistort points prior to mapping them to the ground plane. The errors directly influence, how accurately the location of a tracked person can be mapped from the camera images to the ground plane. As the errors are smaller than the error of the manually measured marker positions (± 50 mm), the homography matrices H_3 and H_7 enable a sufficiently accurate mapping from the camera images to the factory ground plane.

5.7.6.3 Validation via Independent Point Set

To ensure a high calibration accuracy for positions different from those of the calibration markers, the reprojection error is computed for an independent set of validation points. These validation points are acquired by placing a validation target at 5 different locations $V_0, V_1, \ldots, V_4$ on the ground plane. This is shown for the first two positions in fig. 5.14. The marker position is measured both in world coordinates

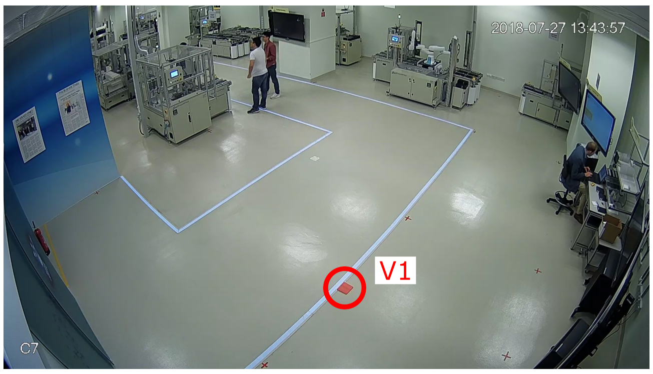
Tab. 5.6: Measured and projected world positions of the extrinsic calibration markers of both camera as well as the according error, which is computed as the euclidean distance between the measured and projected point coordinates.

Camera	Marker	Measured Position		Projected Position		Error
		X/mm	Y/mm	X/mm	Y/mm	e/mm
C_7	P_0	0	0	0.881	0.633	1.085
	P_1	0	3525	−4.591	3527.030	5.018
	P_2	0	5065	3.992	5065.670	4.049
	P_3	2200	−3030	2201.720	−3028.560	2.244
	P_4	2200	0	2197.560	−4.249	4.900
	P_5	2200	3525	2200.550	3525.930	1.076
	P_6	2200	6540	2199.900	6538.710	1.299
	P_7	6940	−200	7027.990	−191.035	88.443
	P_8	6940	3525	6948.910	3603.220	78.729
	P_9	6940	6540	6966.970	6608.920	74.007
C_3	P_7	6940	−200	6943.410	−202.192	4.057
	P_8	6940	3525	6929.530	3521.950	10.900
	P_9	6940	6540	6949.320	6537.830	9.568
	P_{10}	11940	−200	11925.600	−196.669	14.752
	P_{11}	11940	2150	11927.500	2153.68	13.039
	P_{12}	11940	5750	11934.700	5747.070	6.079
	P_{13}	15350	−200	15361.000	−194.677	12.242
	P_{14}	15350	2150	15384.000	2160.230	35.505
	P_{15}	20150	−200	20146.000	−213.121	13.711
	P_{16}	20150	2150	20138.300	2151.170	11.725

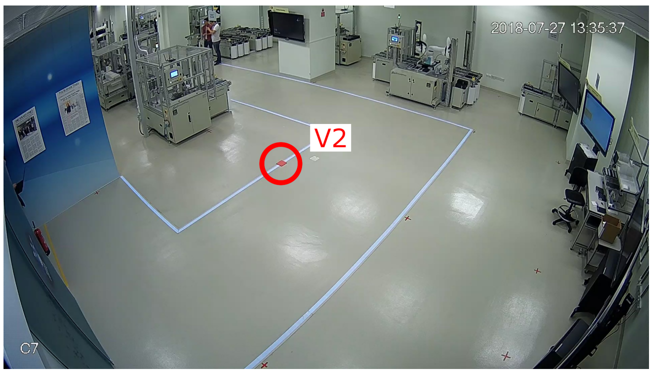
(X, Y) as well as in distorted image coordinates (u, ν) (see table 5.7). The image points are undistorted, yielding the undistorted image points (u', ν'), which are projected onto the ground plane by means of the estimated homography matrix H_7. The euclidean distance between these reprojected points and the measured world positions is a measure of the calibration error. Table 5.8 shows the measured and estimated positions as well as the resulting error for all five validation points. Mean and variance of all 5 errors are 27.949 mm and 21.682 mm respectively. These errors are not larger than the errors at the positions of the calibration markers (see table 5.6). Thus, the calibration parameters can be used to map any image point onto the ground plane with an accuracy sufficient for the intended use case of the Shopfloor Monitor.

Tab. 5.7: Positions of the validation marker on the ground plane, in distorted image coordinates and undistorted image coordinates.

Marker	Ground Plane		Distorted Image		Undistorted Image	
	X/mm	Y/mm	u/pix.	ν/pix.	u'/pix.	ν'/pix.
V_0	1200	0	1516	416	1327.110	439.595
V_1	2200	5125	1015	839	1018.760	684.753
V_2	4700	3525	826	468	914.421	470.297
V_3	4700	6540	417	732	632.638	643.638
V_4	5000	0	1085	282	1056.390	368.053



(a) Validation marker at position V_1.



(b) Validation marker at position V_2.

Fig. 5.14: Two, of the in total 5, example images for validation of the camera calibration results. The validation marker is a pink sponge with dimensions 10 cm *times* 10 cm, that is placed at different locations in the viewport of camera C_7.

Tab. 5.8: Measured and projected world positions of the validation markers as well as the according error, which is computed as the euclidean distance between the measured and projected point coordinates.

Marker	**Measured Position**		**Projected Position**		**Error**
	X/mm	Y/mm	X/mm	Y/mm	e/mm
V_0	1200	0	1197.530	−11.856	12.112
V_1	2200	5125	2195.750	5122.580	4.889
V_2	4700	3525	4727.700	3534.860	29.406
V_3	4700	6540	4714.540	6561.400	25.872
V_4	5000	0	5044.210	50.958	67.464

5.7.7 Calibration Interval

Camera recalibration is only needed, when position, orientation and focal lengths of the cameras change. As these parameters are usually constant, it is sufficient to calibrate the cameras only once during setup of the Shopfloor Monitor. If PTZ cameras are used, the calibration is only valid for a certain orientation and zoom level, which can be stored in a preset. The camera can be moved back into the calibrated position anytime via this preset superseding a recalibration. Recalibrating the cameras might become necessary after some time due to exposure to other environmental factors, for example vibration or temperature differences. Continuous monitoring of the reprojection error allows to determine, if the camera is still properly calibrated or whether a recalibration is necessary.

5.8 Future Improvements

This section describes possible improvements of the current Shopfloor Monitor. Processing speed can be drastically increased by utilizing all 12 available CPU cores to run the update and prediction steps of the core trackers. The current single-core implementation limits the frame rate and prevents the use of computationally more expensive, but more accurate KCF and CSRT core trackers. Only the fast MOSSE tracker can be used. Similarly, the object detector can be sped up by running inference on both available GPUs instead of only a single GPU. Additionally, Nvidia TensorRT [138] can be used to speed up inference by optimizing the inference graph and reducing the precision of numerical operations to float 16 bit. If inference is faster, computationally more complex, but also more accurate object detection algorithms can be used.

To increase detection and tracking accuracy the object detector can be fine-tuned on an application-specific dataset and the single-target trackers can be replaced with a true multi-target tracker. Leveraging appearance features in addition to the bounding box IoU to match detections and existing trajectories can further improve the tracking accuracy.

An issue of the detection and tracking framework is the redundant detection of the same physical target, when this target occurs in more than one camera view. In future, such detections should be merged, if their spatial distance in world coordinates is small and if the extracted appearance features are similar.

Another related problem is the missing handling of target handovers between adjacent frames. As soon as a target leaves one camera view and enters another, its related target ID changes. A solution would be to store a fingerprint for every target in a database and perform a lookup whenever a new target appears. If the target is known, the previous target ID can be applied. This strategy also allows to re-identify targets after a long time and built up target-specific statistics.

Further improvements are possible with regard to the visualization app. Rendering of the floorplan can be sped up by using *buffer geometries* instead of normal geometries. Additional information for each target can be displayed, for example the target ID, a speed vector, the source camera or the elapsed time since the first detection. The visualization app can also be extended with basic analysis features, for example a heatmap plot of the location and speed distributions of targets over different time periods. To enable fast deployment of the Shopfloor Monitor in another factory, the model file of the floorplan should be created automatically from a CAD model.

Apart from that, the camera calibration provides improvement potential as well. For example, measuring positions of the extrinsic calibration markers with smaller errors than the current ± 50 mm, can improve the results of the camera calibration and the mapping accuracy from image to world coordinates. Furthermore, automating the calibration procedure with one of the methods in [139–141] enables easier and faster deployment of the Shopfloor Monitor in another factory.

5.9 Conclusion

In this chapter the Shopfloor Monitor was implemented. After a description of the overall system architecture, individual components, such as the main server process, the MySQL database and the client web apps were explained. The main process handles the acquisition of the video streams, run the detection and tracking algorithms, maps from image to world coordinates, sends the tracking results over websockets to the client web app and inserts it into a database. The next section expanded on the detection and tracking framework. This framework tracks targets by means of multiple single-target trackers, which are updated in regular intervals based on the results of an object detector. Afterwards, the layout of the MySQL database for long-term storage of the tracking results and a general framework for client web apps were presented. Finally, a web app for real-time visualization of targets on a 2D floorplan of the factory was introduced. The second part of this chapter focused on the camera calibration. First, a parametric model of the fisheye camera was introduced. The model parameters were estimated from point correspondences between images and the observed scene and allowed to undistort the camera images. Homographies between the image planes and the factory ground plane were estimated and used to map target locations from undistorted image to world coordinates. Finally, camera calibration was evaluated by computing reprojection errors and errors on a set of independent validation markers. The chapter concluded with an overview of possible future improvements. In the next chapter experiments are conducted to analyze different hyperparameter configurations of the Shopfloor Monitor.

Chapter 6

Experiments

The Shopfloor Monitor, being a detection and tracking system at its core, contains a variety of different hyperparameters (see chapter 5) which control the tracking behaviour of the system. Understanding the influence of each parameter on the tracking performance helps to choose suitable values for the parameters, which yield a good trade-off between the accuracy of estimated trajectories and the processing frame rate. It also helps to determine how critical each individual parameter is for the performance, and thus can give direction to further improvements of the system. Insensitivity of tracking performance to certain parameters might also indicate a faulty implementation or an unnecessary feature of the tracking system, which can be removed without negatively affecting performance. As opposed to often conducted ablation studies [37, 142–145], in which individual features are removed to determine their impact on the performance, a parameter studies provides deeper insights as a feature is not simply removed, but its impact modulated by choosing a variety of parameter values.

The aim of the study conducted in this chapter is not to exhaustively evaluate all possible parameter combinations in form of a grid search or randomized search to find an optimal parameter combination, but rather lies in gaining an understanding of the influence of different parameters on the tracking performance. To conduct this study, an evaluation tool is developed that enables an automated execution of all experiments and reporting of the results and allows to evaluate features added in future versions of the monitoring system.

This chapter first introduces the dataset specifically created for evaluation of the monitoring system in section 6.1. In section 6.2 the utilized performance metrics are explained. The parameters under study and the experimental design are described in section 6.3. Experimental results are presented in section 6.4. Finally, section 6.5 shows qualitative results of the detection and tracking system.

6.1 Evaluation Dataset

Basis of the evaluation of the Shopfloor Monitor is a video dataset containing annotated ground-truth bounding boxes for every person in every frame. There are several public datasets for visual object tracking available, for example the Multi Object Tracking (MOT) dataset [7] or the Visual Object Tracking (VOT) dataset [146]. However, as the aim of this evaluation is to understand the influence of the system parameters under the given operating conditions in the factory, only a task-specific evaluation dataset, based on the video footage acquired by the monitoring system itself guarantees transferability of the analysis results, and therefore such a dataset is created and described in this section.

The evaluation dataset contains five video sequences with a resolution of 1920×1080 pixels, a frame rate of 15 FPS and an average duration of 15.4 s. Further details can be found in table 6.1. Additionally, table 6.2 shows statistics about the width, height and aspect ratio of all bounding boxes in the dataset. These sequences are manually extracted from a set of 22 videos with a total duration of 4 hours and 91 seconds, recorded by the two cameras used in the monitoring system (see section 5.6) over the course of one day. To simplify the analysis task and not consider any issues introduced through a multi-camera setup, only recordings of a single camera C_7 are considered. Furthermore, the sequences are continuous and do not contain any cuts or abrupt scene changes. They are chosen based on criteria, such as high number of people with varying activities (standing, walking or sitting), crowded scenes, occlusion events and people entering or leaving the scene to provide a challenging and versatile test environment for the tracking system, which helps to unveil weaknesses of the developed approach and retrieve rich analysis results.

Tab. 6.1: Descriptive statistics of the evaluation dataset. IDs are the number of unique people in each sequence, whereby entries and exits count the number of people entering and exiting the scene throughout the sequence. Density is the average number of bounding boxes per frame.

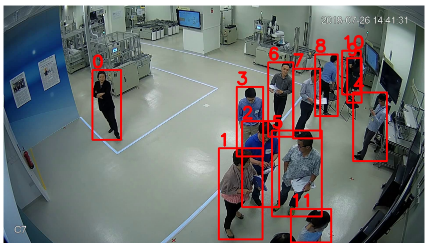
Name	**Resolution**	**FPS**	**Length**	**IDs**	**Boxes**	**Density**	**Entries**	**Exits**	**Description**
Seq-0	1920 × 1080	15	226 (00:15)	18	3575	15.82	4	2	Crowd sitting + few people walking and standing
Seq-1	1920 × 1080	15	120 (00:08)	13	1491	12.43	0	1	Crowd slowly walking
Seq-2	1920 × 1080	15	175 (00:11)	19	2913	16.65	3	4	Crowd standing + few people walking
Seq-3	1920 × 1080	15	188 (00:12)	17	2820	15.00	3	6	Large number of targets walking
Seq-4	1920 × 1080	15	178 (00:11)	10	1393	7.83	1	4	Medium number of targets walking
Total	–	–	888 (00:57)	77	12192	13.73	11	17	-

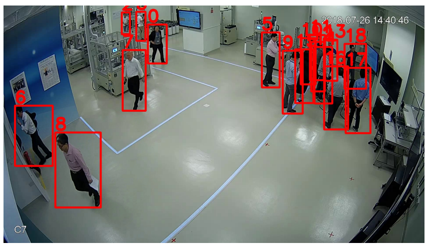
Tab. 6.2: Advanced statistics of all bounding boxes in the evaluation sequences.

Name	**Box Width**			**Box Height**			**Box Aspect Ratio**		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Seq-0	35.40	146.74	328.20	159.89	318.59	400.00	0.09	0.46	2.05
Seq-1	60.80	128.03	258.87	42.50	278.09	450.80	0.13	0.46	6.09
Seq-2	21.85	88.33	293.30	55.72	227.95	417.82	0.05	0.39	5.26
Seq-3	40.80	153.41	345.85	84.66	284.30	451.53	0.09	0.54	4.09
Seq-4	51.00	141.87	282.10	167.73	303.69	415.53	0.12	0.47	1.68
Total	21.85	131.676	345.85	42.50	282.524	451.53	0.05	0.096	6.09

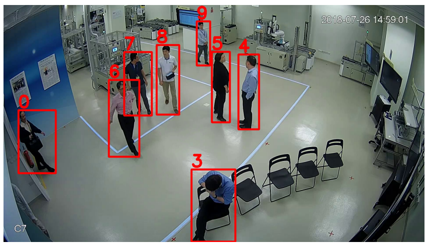
After extraction of the five sequences, ground-truth bounding boxes and IDs of every person in every frame are annotated with the help of the Computer Vision Annotation Tool (CVAT) [147] and exported as XML files. This tool speeds up the annotation task by interpolating bounding box coordinates in all frames between a lower number of manually annotated key frames. During annotation some rules according to [7] are followed to ensure consistent annotation of all frames. First, bounding boxes are only drawn around humans. Every bounding box should envelope the entire person as tightly as possible, whereby no body parts should protrude over the box edge. Clothing is counted as part of the person, but not so any other accessories, such as phones, backpacks or notebooks. Each person is fully annotated through occlusion as long as the body shape can be determined accurately. Very small people in the far background of the scene as well as permanently occluded people are not annotated. People entering the scene are annotated as soon as ca. 50 % of their body is visible. Accordingly, people leaving the scene are marked as *out-of-view* when half of their body lies outside of the frame. In such cases bounding boxes are not allowed to protrude over the frame edge and are cropped instead. Finally, people re-entering the scene are always given a new ID. Some example frames annotated this way are shown in fig. 6.1.

(a) Seq-0 frame 40.


(b) Seq-1 frame 98.


(c) Seq-2 frame 141.

(d) Seq-3 frame 71.


(e) Seq-4 frame 57.

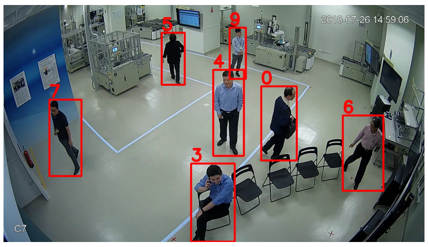

(f) Seq-4 frame 136.

Fig. 6.1: Examples for annotated ground-truth bounding boxes and IDs in the five evaluation sequences.

6.2 Evaluation Metrics

In recent years a wide variety of performance metrics for object tracking has been presented. Several of those metrics focus on single target tracking [146, 148], while others are used to assess performance of multi-target trackers [149–161]. Some research provides application-specific metrics for video surveillance [162–168], pedestrian tracking [169] or vehicle tracking [170]. More recent research provides benchmarks with small numbers of descriptive performance metrics enabling a quantitative ranking of different tracking algorithms. Examples for such benchmarks are the Visual Object Tracking (VOT) Challenge [146], which uses the accuracy and robustness metrics from [148], the CLEAR challenge [171], and the Multiple Object Tracking (MOT) Benchmark [7]. The following section briefly introduces the metrics included in the MOT benchmark, which are used to evaluate the developed tracking framework on the previously described evaluation dataset. As can be seen in table 6.3, the MOT metrics comprise standard metrics, such as precision and recall, the two CLEAR metrics multiple object tracking accuracy (MOTA) and multiple object tracking precision (MOTP) [152] and newer ID metrics proposed in [161]. To ensure correctness of the results, the official Python MOTMETRICS package [172] is used to compute all metrics.

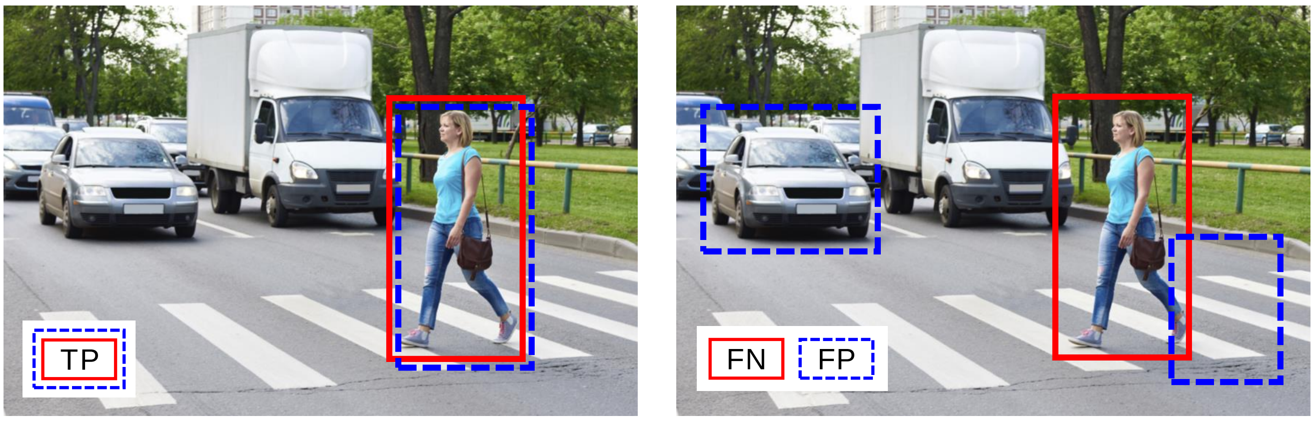
All performance metrics are computed by comparing the set of bounding boxes predicted by the Shopfloor Monitor and the set of manually annotated ground-truth bounding boxes in each frame of the evaluation sequences. In literature, a ground-truth bounding box is often called the *target*, while the according predicted bounding box is a *hypothesis* of the tracker [7]. A well-performing tracker is able to predict bounding boxes which mostly overlap with the according ground-truth boxes for most of the targets in each frame. Such a match between hypothesis and ground-truth is a true positive (TP). In the MOT benchmark a match is counted when the intersection over union (IoU) (see appendix A.3) of the predicted and ground-truth bounding box is larger than a threshold of 50 %. This is shown in fig. 6.2. If the IoU is smaller than this threshold or there is no hypothesis at all for a target, the target is called a miss or false negative (FN). If a hypothesis can not be matched to any target, it is counted as a false positive (FP). These measures can be computed frame by frame as they do not take any temporal dependencies between the target locations into account [151, 158]. They allow computation of the precision (PRCN)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{6.1}$$

which quantifies how many of the predictions made by the tracker are matches. Furthermore, the recall (RCLL)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.2}$$

can be defined as the fraction of all correctly matched bounding boxes as compared to the total number of ground-truth bounding boxes [159].



(a) A true positive (TP). **(b)** A false negative (FN) and two false positives (FP).

Fig. 6.2: Examples of a true positive, a false negative and two false positives. A true positive is a match of a tracker hypothesis (dashed blue box) with a ground-truth target (solid red box), while a false positive is a hypothesis, which could not be matched to any ground-truth target, and a false negative is a ground-truth which could not be matched with any hypothesis. A match is established if the IoU of the hypothesis and ground-truth box is larger than 0.5. [151, 158] (Background image from [173])

Apart from these simple frame-based metrics, the MOT metrics comprise metrics specifically designed for tracking, which consider temporal information about the association between predictions and ground-truth targets as well as target IDs [7]. To compute these measures, again each hypothesis has to be matched to its according ground-truth target. This time, the matches are computed in a way to maximize the total IoU of all predicted and ground-truth bounding boxes in all frames while simultaneously keeping track of the matched IDs and preferring previously matched IDs for forming new matches [152]. This way, IDs are kept as constant as possible. The ID switch metric (IDSW) measures how often ID switches occur in a sequence. Whenever a hypothesis is matched to a ground-truth target and the ID of the target differs from the one in the previous frame, an ID switch is counted (see fig. 6.3) [7].

The multiple object tracking accuracy (MOTA)

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDSW}_t)}{\sum_t \text{GT}_t} \tag{6.3}$$

is an intuitive measure for all errors made by the tracker, such as misses, false positives and ID switches in relation to the total number GT_t of ground-truth targets in each frame. The MOTA is computed in percent and can lie in the range $(-\infty, 100]$, whereby it gets negative when the number of errors exceeds the total number of targets in a sequence [7, 152]. Similarly, the multiple object tracking precision (MOTP)

$$\text{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t \text{TP}_t}. \tag{6.4}$$

provides an intuitive measure for the average overlap of all matched bounding boxes and lies within the range of 0 % to 100 %. In the equation, $d_{t,i}$ is the IoU of the ith hypothesis and the according ground-truth bounding box and TP_t are the true positives in frame t [7, 152].

Additional metrics in the MOT benchmark classify each ground-truth trajectory as mostly tracked (MT), partly tracked (PT) or mostly lost (ML) based on the proportion of frames in which the target belonging to this trajectory is properly tracked, meaning that a hypothesis with an IoU of more than 50 % is provided [7, 160]. See fig. 6.3 for a visualization of this. If a target is tracked in more than 80 % of the frames belonging to that trajectory, it is classified as MT. If it is tracked for less than 80 % and more than 20 % of its length, a trajectory is classified as PT, and if it is tracked for less than 20 % of frames, it is categorized as ML. ID switches are not considered in the computation of these measures. An additional measure is the fragmentation (FM), which indicates the total number of gaps in each tracked ground-truth trajectory. A gap occurs when a target that was tracked properly in the previous frame is lost in the new frame and re-identified in one of the future frames [7].

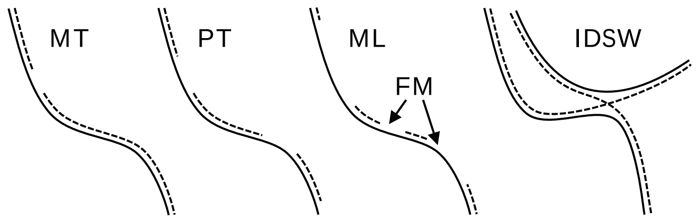
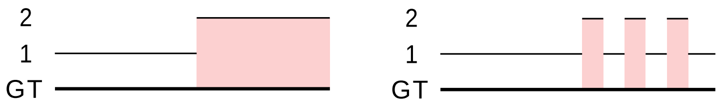


Fig. 6.3: Visualization of mostly tracked (MT), partly tracked (PT) and mostly lost (ML) ground-truth trajectories (solid lines). An ID switch (IDSW) and fragmentations (FM) are also shown. [7, 160]

The last set of evaluation metrics in the MOT benchmark are the ID metrics proposed in [161]. They comprise the ID false positive (IDFP), ID false negative (IDFN), ID true positive (IDTP) and allow computation of ID precision (IDP) and ID recall (IDR) equal to eq. (6.1) and eq. (6.2). The IDF_1 measure

$$\mathrm{IDF}_1 = \frac{2\mathrm{IDTP}}{2\mathrm{IDTP} + \mathrm{IDFP} + \mathrm{IDFN}} \tag{6.5}$$

is the harmonic mean of ID recall and ID precision and thus provides a balanced view of both measures. As opposed to their non-ID counterparts, ID metrics do not count how often certain events, such as a miss, false positive or ID switch, occur, but instead they describe over which portion of a sequence the identity of a target is correctly tracked. This is done by jointly matching exactly one ground-truth trajectory to every hypothesis trajectory in a way that maximizes the total number of frames in which the matched ground-truth and hypothesis IDs remain unchanged. To clarify the difference, consider a sequence of 10 frames with a single ID switch in the 5th frame. The original IDSW metric would count only a single ID switch despite the target being falsely identified over half of the sequence. The ID precision would yield the more intuitive value of 50 %. For another example see fig. 6.4. However, depending on the application, both event based as well as ID-based metrics allow useful insights into the performance and possible weaknesses of a tracker.



(a) IDSW: 1, IDP: 50 %　**(b)** IDSW: 6, IDP: 75 %

Fig. 6.4: Example showing the difference between the IDSW and IDP metric. IDSW counts the number of ID switches, while IDP states the ratio of frames, in which ID 1 is correctly assigned to the ground-truth trjectory. [161]

A last metric, not included in the MOT benchmark, but of high interest for the task at hand, is the frame rate at which the detection and tracking system can process frames of an incoming video stream. The momentary frame rate f_k of a video sequence at frame t_k is computed as

$$f_k = \frac{1}{t_{k+1} - t_k}, \quad k = \{0, 1, \dots, n-1\}, \tag{6.6}$$

whereby usually the mean over all n frames of the sequence is computed, yielding a scalar value (FPS) for the frame rate.

Apart from the presented evaluation metrics, there exist further metrics, such as the false alarm rate (FAR), false negative rate (FNR), F_1-score [158, 159], multi-camera object tracking accuracy (MCTA) [174], time lag, object localization metrics and occlusion success rate (OSR) [159]. Additionally, various performance curves can be created by changing hyperparameters of the evaluation metrics, such as the IoU threshold or location error threshold during matching. Examples are the precision-recall-curve [159], the overlap-sucess-curve [39], showing sucess rate over the IoU threshold, or the precision-distance-curve, which plots precision over the location error threshold [39]. Furthermore, it is possible to compute all presented performance metrics not only in image coordinates, but also in world coordinates after applying the according mapping (see section 5.7.5) to tracked image locations. However, these additional performance metrics are not used to simplify the analysis task and obtain easily interpretable results in the following experiments.

Tab. 6.3: Overview of the utilized MOT evaluation metrics [7, 161]. Desired optimal values are in bold. Note however, that MT, PT and ML are dependent on each other.

Category	**Metric**	**Symbol**	**Range**	**Unit**
Detection Based	True Positive	TP	$[0, \boldsymbol{\infty}]$	-
	False Negative (Miss)	FN	$[\mathbf{0}, \infty]$	-
	False Positive	FP	$[\mathbf{0}, \infty]$	-
	Precision	PRCN	$[0, \mathbf{100}]$	%
	Recall	RCLL	$[0, \mathbf{100}]$	%
Tracking Based	ID Switch	IDSW	$[\mathbf{0}, \infty]$	-
	Multiple Object Tracking Accuracy	MOTA	$(-\infty, \mathbf{100}]$	%
	Multiple Object Tracking Precision	MOTP	$[0, \mathbf{100}]$	%
	Mostly Tracked	MT	$[0, \boldsymbol{\infty}]$	-
	Partly Tracked	PT	$[\mathbf{0}, \infty]$	-
	Mostly Lost	ML	$[\mathbf{0}, \infty]$	-
	Fragmentation	FM	$[\mathbf{0}, \infty]$	-
ID Based	ID True Positive	IDTP	$[0, \boldsymbol{\infty}]$	-
	ID False Negative	IDFN	$[\mathbf{0}, \infty]$	-
	ID False Positive	IDFP	$[\mathbf{0}, \infty]$	-
	ID Precision	IDP	$[0, \mathbf{100}]$	%
	ID Recall	IDR	$[0, \mathbf{100}]$	%
	ID F_1-Score	IDF_1	$[0, \mathbf{100}]$	%
Speed	Frame Rate	FPS	$[0, \boldsymbol{\infty}]$	Hz

6.3 Design of Experiments

In this section a suitable evaluation strategy for the detection and tracking system is developed, with the aim to provide an understanding of the connection between system parameters and the overall tracking performance.

The parameters under consideration are summarized in table 6.4. Apart from these, there are two fixed parameters, the category filter, which filters out all non-person detections, and the box size filter, which removes all boxes larger than 0.25 times the frame width and 0.6 times the frame height. As these parameters are temporal solutions, which are likely to be removed in future versions of the monitoring system, they are not of interest for the parameter study and thus excluded. All other parameters are explained in detail in chapter 5, however, shall be briefly recapitulated. The two most obvious parameters are the object detector $\mathcal{A}_{\text{det}}$ and tracker $\mathcal{A}_{\text{trk}}$, which are both utilized to predict bounding boxes around humans in a scene. There are many different object detectors and trackers available (see appendix A and appendix B). Thus, a small subset of seven different object detectors and three tracking algorithms are chosen for analysis. Five of the detectors are Faster R-CNN models [9] with different base networks [175–177], one is a R-FCN model [178] and the last one a Single Shot Detector (SSD) [11]. All of these models are readily implemented and pretrained as part of the Tensorflow Object Detection library [119]. As trackers MOSSE [12], KCF [179] and CSRT [13] are used as they are the most up-to-date models in the OpenCV library [49]. The detection cycle frequency f_d of the monitoring system controls how frequently the object detector is run to generate bounding box proposals and update the states of the object trackers. Here, four different values 1, 2, 5 and 10 are chosen, meaning the detector is run either on every frame or on every 2nd, 5th or 10th frame. To reduce the computational cost of processing a single frame, each frame is scaled down by a scaling factor s prior to processing. The effect of this on the tracking performance is evaluated by choosing three different values, 0.4, 0.6 and 1.0, for the scaling factor, whereby 1.0 will disable resizing and feed the tracker with the original 1920×1080 pixel input frame. However, the scaling factor does not effect the size of input frames to the detector, which are resized to a fixed size of either 300×300 pixels in case of SSD or 1024×600 pixels in case of Faster R-CNN and R-FCN detectors. Another parameter of the tracking system is the match IoU threshold θ_{IoU}, which specifies the minimum IoU needed to match a bounding box, proposed by the detector every f_d frames, with the bounding box of a tracker. On a successful match the tracker state is updated accordingly. By default this IoU threshold is set to 30 %, but values of 10 %, 50 %, 70 % and 90 % are analyzed, too. The last parameter observed in this study is a pair of two natural numbers θ_v and θ_d, named the hypothesis valid and hypothesis deletion threshold, which are subsequently set to values $(0, 0)$, $(2, 2)$, $(5, 5)$, $(5, 0)$ and $(0, 5)$. The first parameter specifies the minimum number of subsequent frames in which a target hypothesis needs to be confirmed by the object detector in order to mark this target as valid and create a bounding box. The second threshold determines for how many subsequent frames a target might be missed by the object detector before its related bounding box and tracker are deleted.

A full setup of the Shopfloor Monitor requires a valid value for each of the six parameters. By default a Faster R-CNN/ResNet-101 detector is used in combination with a MOSSE tracker, a cycle frequency of $f_d = 2$, a scaling factor $s = 0.4$, an IoU threshold $\theta_{\text{IoU}} = 0.3$ and hypothesis valid and deletion thresholds of $\theta_v = 5$ and $\theta_d = 5$. To analyze the influence of each parameter on the tracking performance, the value of a single parameter is changed at a time while all other parameters are set to their default values. This way of alternating parameter values yields a relatively low and easy-to-handle number of 27 different configurations of the monitoring system, whereby six of those are redundant and equal to the default configuration. After each change the whole set of evaluation metrics introduced in section 6.2 is computed for each of the five evaluation sequences created in section 6.1. The exact procedure of this is shown in algorithm 7. First, all six parameters and the according values are defined along the five evaluation sequences. For each of the parameters a single value is chosen out of the range of possible values, while all other parameters are set to their default values. With the monitoring system setup like this, it is run on each of the five evaluation sequences yielding a set of bounding box hypotheses for each frame of each sequence. Afterwards, the manually annotated ground-truth bounding boxes for each frame are loaded, and an accumulator object of the MOTMETRICS package, which is needed for computation of the evaluation metrics, is iteratively filled with the set of ground-truth bounding boxes and the according hypotheses of each frame in the sequence. Here, only hypotheses which are marked as valid are considered.

Tab. 6.4: Description of the studied parameters of the Shopfloor Monitor. For a more in-depth explanation of the meaning of each of those parameters, please refer to chapter 5. Default parameters are in bold. FRCNN stands for Faster R-CNN and the tag *low proposals* (lp) indicates a reduced amount of object proposals. Detectors are given an alias in brackets, which is used in the analysis plots for easier plotting.

Parameter	**Symbol**	**Description**	**Values**
Detector	$\mathcal{A}_{\text{det}}$	Object detector and base architecture used for detection of targets every f_d frames	FRCNN/NASNet (Det-0) FRCNN/Inception v2 (Det-1) **FRCNN/ResNet-101 (Det-2)** FRCNN/ResNet-101 (lp) (Det-3) FRCNN/ResNet-50 (Det-4) RFCN/ResNet-101 (Det-5) SSD/ResNet-50 (Det-6)
Tracker	$\mathcal{A}_{\text{trk}}$	Tracker used to track targets in intermediate frames	**MOSSE**, KCF, CSRT
Cycle Frequency	f_d	The detector is run every f_d frames	1, **2**, 5, 10
Scaling Factor	s	Each frame is resized with this factor previous to processing	**0.4**, 0.6, 1.0
Match IoU Threshold	θ_{IoU}	Minimum IoU needed for a match between a tracked target and detections made every f_d frames	0.1, **0.3**, 0.5, 0.7, 0.9
Hypothesis Thresholds	θ_v, θ_d	Minimum number of matches needed before a tracking hypothesis is valid and maximum number of misses allowed before a hypothesis is deleted	$(0,0)$, $(2,2)$, $\mathbf{(5,5)}$, $(5,0)$, $(0,5)$

Finally, the accumulator is appended to a list. All five accumulators for a tracker configuration are then passed to a function which computes and returns the evaluation metrics for logging. In addition to the sequence-specific metrics, an average over all evaluation sequences is computed. In the following analysis all stated metrics are such averages to reduce the amount of data and increase the robustness of results to varying conditions in the evaluation sequences, such as a varying number of targets.

It is important to note that the analyzed detection and tracking system is entirely deterministic, yielding the same results for all evaluation metrics when rerunning the described experiments. Thus, it is sufficient to run all experiments only once and state a single scalar value for each evaluation metric, rather than computing a distribution of results. The only exception is the frame rate, which varies depending on how the operating system schedules execution of the program. Hence, it would be necessary to rerun experiments multiple times to compute a distribution of frame rates. However, computation time of the analysis increases linearly with the number of runs, which is why it is desirable to compute results for only a single run. As the standard deviation of the frame rate within the six redundant configurations is only 0.532 Hz, and thus relatively small compared to the variations of the frame rate between different experiments, waiving the statistical nature of the frame rate and running the experiments only once provides still useful, even though statistically not fully described results.

Although the detection and tracking system provides target locations both in image as well as in world coordinates, evaluation metrics are solely computed based on the target locations in image coordinates. All presented evaluation metrics could be easily computed in world coordinates as well, by replacing the IoU-based distance metric for target association by another point-based distance metric, such as the Euclidean distance. However, as manual annotation of targets is performed in the image domain, it would be necessary to transform the ground-truth annotations from image to world coordinates by

Algorithm 7: Computation of evaluation metrics.

```
Data: parameters, values, evaluation_sequences
  for each param in parameters do
      for each value in values[param] do
          accumulators ← [ ]
          for each seq in evaluation_sequences do
              hyp_boxes ← RUN_TRACKER(seq)
              gt_boxes ← LOAD_GROUND_TRUTH_BOXES(seq)
              acc ← new ACCUMULATOR()
              for each frame in seq do
                  acc.UPDATE(hyp_boxes[frame], gt_boxes[frame])
              end for
              accumulators[seq] ← acc
          end for
          metrics ← COMPUTE_METRICS(accumulators)
          print param, value, metrics
      end for
  end for
```

applying the according transformation (see section 5.7.5). Since the same mapping is also performed in the tracking software to compute world from image coordinates, skipping the coordinate transformation and computing metrics directly on image locations yield the same results. This argumentation holds not true, though, if advanced functions, such as merging of target occurrences across multiple camera views, are considered. In this case, multiple image coordinates are mapped to the same world coordinate, and thus evaluation metrics in image and world domain would differ. However, the evaluation follows a multi-target single-camera approach wherefore this advanced case does not apply and computation of evaluation metrics in image coordinates is sufficient.

6.4 Analysis Results

Running the experiments designed in the previous section, yields the 27 sets of evaluation metrics shown in the rows of table 6.5. Here, the first two columns specify the parameter and its according value under analysis. The remaining parameter values are kept at their defaults, resulting in a full parameter configuration for each row of the table. The rest of the columns reports the resulting evaluation metrics for the given parameter configuration averaged over the results computed on the five individual evaluation sequences. The full results, including those for the different evaluation sequences, can be found in appendix C.2.

To give a more lively illustration of the analysis results, all values are plotted as stemplots in fig. 6.5, fig. 6.6 and fig. 6.7, whereby each diagram covers a portion of the evaluation metrics. The horizontal axes show the 27 discrete parameter configurations, whereas the vertical axes displays the continuous evaluation metrics. Coloring of the stems indicates which one of the six parameters, detector, tracker, cycle frequency, scaling factor, match IoU threshold or hypothesis thresholds, is modulated. This is also shown by the horizontal axis at the bottom of each set of plots. The number of stems per group relates to to the number of possible values for each parameter. As an example, the second stem on the left of each plot indicates the resulting metric for a configuration of the monitoring system with all parameter values set to their defaults apart from the detector, which is a Faster R-CNN/Inception-ResNet v2 (Det-1) model (see table 6.4). The stems plotted in black correspond to the six redundant configurations equal to

Tab. 6.5: Analysis results showing the tracking performance at different parameter values. The single parameter, which is varied, is stated alongside its value in the first two columns. All other parameters are set to their defaults, as mentioned in the text. For specification of the detection algorithms refer to table 6.4. Units of the metrics are according to table 6.3. The values in bold state the best result for a metric over all possible values of a parameter.

Param.	**Value**	IDF$_1$	IDP	IDR	RCLL	PRCN	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Detector	Det-0	70.0	76.3	64.6	**74.5**	87.9	36	35	**6**	1248	**3115**	50	**78**	63.8	19.0	53.19
	Det-1	**73.7**	**83.1**	66.2	73.6	**92.4**	**37**	33	7	**741**	3217	36	86	**67.2**	20.0	56.18
	Det-2	73.4	81.6	**66.7**	72.7	89.0	36	34	7	1095	3332	**33**	93	63.4	22.0	58.08
	Det-3	67.4	79.7	58.4	67.0	91.5	26	38	13	756	4021	37	106	60.5	21.9	**61.59**
	Det-4	66.5	77.3	58.3	67.3	89.3	30	34	13	987	3987	47	106	58.8	**23.6**	58.40
	Det-5	66.5	72.1	61.7	69.3	81.1	36	29	12	1975	3737	39	128	52.8	**23.6**	52.68
	Det-6	56.4	70.4	47.0	57.2	85.7	22	35	20	1160	5213	45	79	47.4	22.2	61.24
Tracker	MOSSE	73.4	**81.6**	66.7	72.7	89.0	36	34	**7**	1095	3332	**33**	93	63.4	**22.0**	**58.18**
	KCF	73.1	81.1	66.5	73.4	89.5	40	29	8	1052	3241	**33**	**91**	64.5	20.6	10.40
	CSRT	**73.6**	81.1	**67.3**	**74.8**	**90.1**	**41**	29	**7**	**1005**	**3076**	36	100	**66.2**	20.6	3.34
Cycle Frequency	1	**73.4**	79.5	**68.1**	**78.0**	**91.0**	**44**	26	**7**	**936**	**2683**	64	145	**69.8**	21.8	8.83
	2	**73.4**	**81.6**	66.7	72.7	89.0	36	34	**7**	1095	3332	33	93	63.4	22.0	57.55
	5	63.4	77.8	53.6	57.0	82.9	17	44	16	1434	5237	27	60	45.1	22.5	81.80
	10	50.9	79.7	37.4	38.3	81.8	0	41	36	1043	7519	**11**	**27**	29.7	**22.8**	**87.82**
Scaling Factor	0.4	**73.4**	**81.6**	**66.7**	**72.7**	89.0	**36**	34	**7**	1095	**3332**	33	93	**63.4**	22.0	**58.84**
	0.6	69.3	78.1	62.3	70.4	88.2	32	37	8	1144	3613	**31**	108	60.7	22.1	37.30
	1.0	69.9	78.8	62.9	71.2	**89.2**	32	38	**7**	**1046**	3515	33	113	62.3	**22.6**	16.85
Match IoU Threshold	0.1	**76.0**	**83.6**	**69.7**	**74.3**	89.1	**37**	31	9	1109	**3130**	34	107	**65.0**	22.0	**59.48**
	0.3	73.4	81.6	66.7	72.7	89.0	36	34	7	1095	3332	**33**	93	63.4	22.0	57.92
	0.5	66.4	75.5	59.3	69.7	88.8	29	42	**6**	1070	3690	53	81	60.5	22.1	56.63
	0.7	56.4	71.6	46.6	59.5	91.5	24	37	16	673	4936	82	85	53.3	**22.4**	53.21
	0.9	13.7	51.2	7.9	14.7	**95.4**	1	15	61	**86**	10397	47	**46**	13.6	19.2	42.77
Hypothesis Thresholds	$(0,0)$	69.7	74.6	65.5	79.0	90.0	45	25	7	1065	2563	91	158	**69.5**	22.3	61.10
	$(2,2)$	72.5	78.1	67.7	78.1	90.1	46	24	7	1043	2675	63	133	69.0	22.3	59.35
	$(5,5)$	**73.4**	81.6	66.7	72.7	89.0	36	34	7	1095	3332	**33**	**93**	63.4	22.0	59.10
	$(5,0)$	69.2	**83.2**	59.3	66.9	**93.9**	31	38	8	**528**	4032	55	121	62.1	21.8	**61.19**
	$(0,5)$	72.2	72.4	**72.0**	**82.0**	82.4	**51**	21	**5**	2130	**2195**	67	126	64.0	**22.5**	57.35

the default setup. Thus, all metrics of these six configurations, apart from the statistically natured frame rate, are equal. They are only shown for the sake of completeness and to underline the experimental strategy. In the following, the results presented in the table and stem plots will be described in detail for every set of parameter values and important aspects will be highlighted.

6.4.1 Detection Algorithm

The choice of object detector used to update tracker states every f_d frames has a large impact on the evaluation metrics, as the models covered in the analysis exhibit diverse characteristics. Considering the official values of the COCO mAP (see appendix A.3) and inference times of each object detector (see table 6.6), it is shown that the Faster R-CNN models based on NASNet and Inception-ResNet v2 yield superior detection accuracy at the cost of higher inference time, while the other detectors achieve

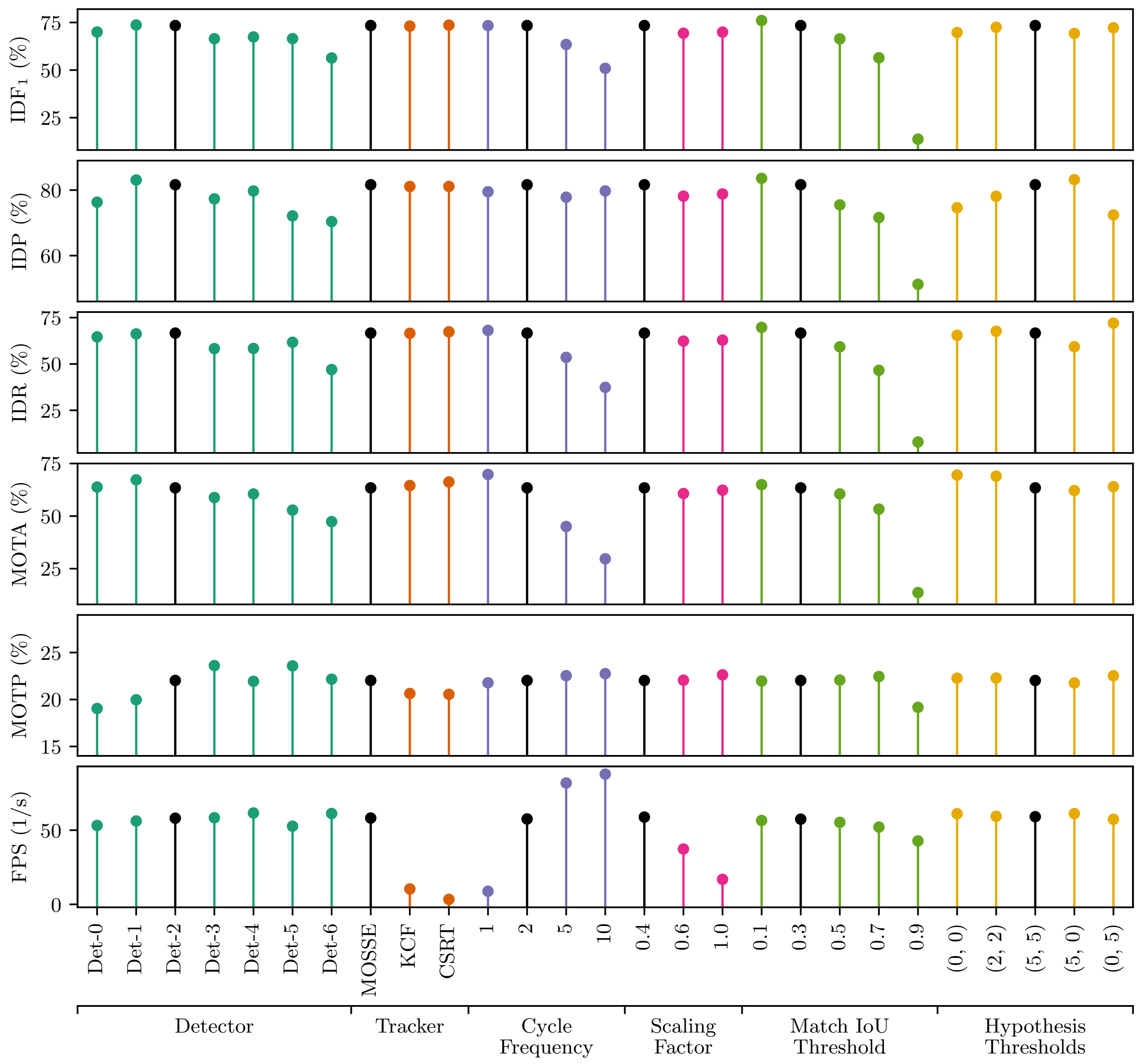


Fig. 6.5: Resulting ID metrics (IDF$_1$, IDP, IDR), MOTA, MOTP and frame rate (FPS) for all tested configurations (see table 6.4). Black stems correspond to the default configuration.

less accurate results, but operate at a higher processing speed. The results of the study mainly meet these expectations, but also reveal some unexpected insights. The larger mAP of the first three analyzed detectors is reflected in higher scores for the IDF$_1$, IDP and IDR metrics, MOTA and recall as well as a lower number of false negatives, fragmentations and ID switches. Moreover, less ground-truth trajectories are mostly lost (ML) and more are mostly tracked (MT). Accordingly, the results for the other four object detectors are slightly worse. The MOTP shows the opposite behavior and is lower for the first three detectors than for the remaining detectors. For the precision and the number of false positives there is no clear pattern discernible, however, here, the R-FCN yields especially bad scores of 81.1 % for the precision and 1975 false positives. This detector also produces the highest number of fragmentations (128) for all tested detectors. However, R-FCN achieves a high number of 36 mostly tracked trajectories, which is almost on the level of the Faster R-CNN/Inception-ResNet v2 model, yielding 37 mostly tracked trajectories. Despite its high mAP of 35, the Single Shot Detector achieves shows

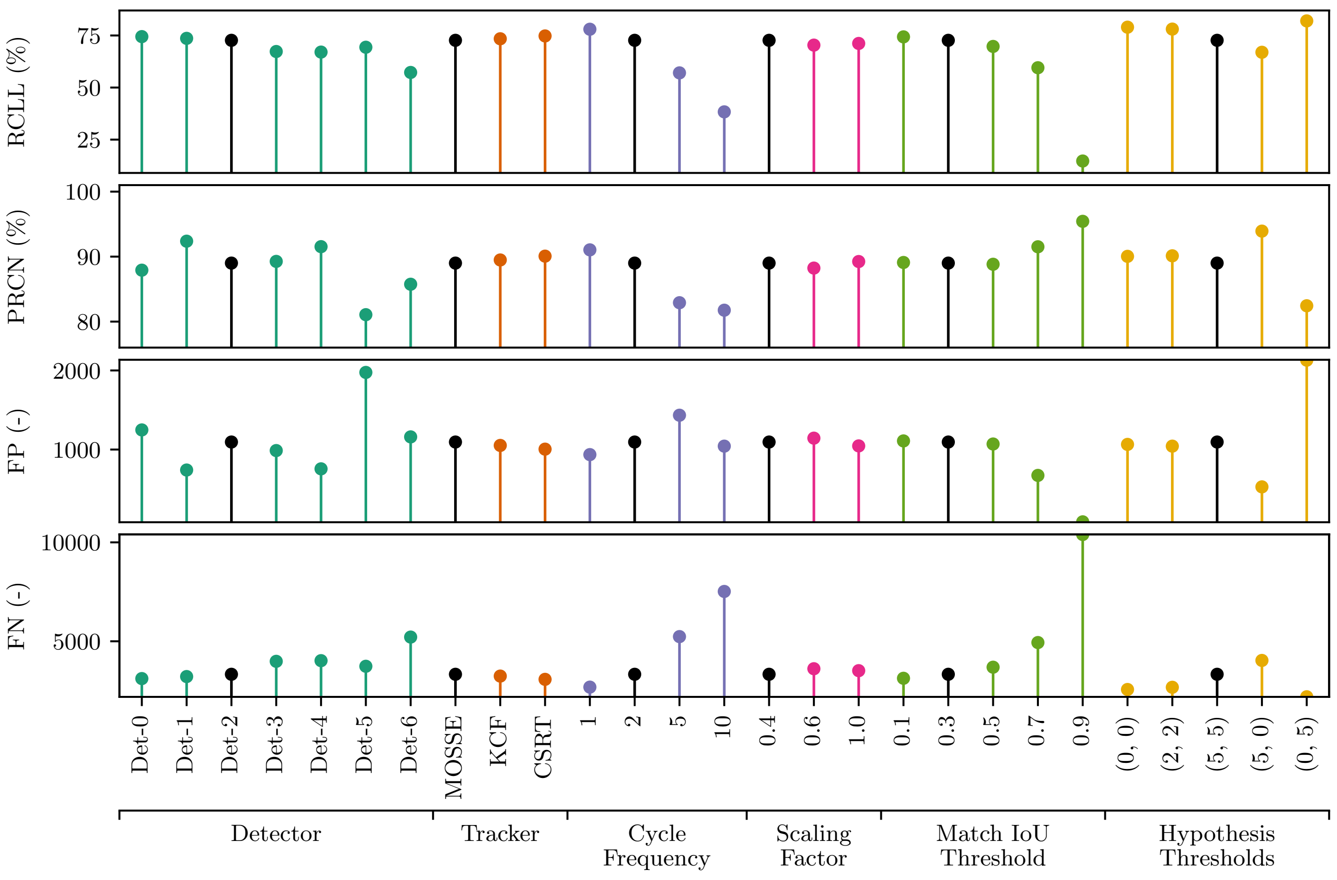


Fig. 6.6: Resulting number of false positives (FP), false negatives (FN), recall (RCLL) and precision (PRCN) for all tested configurations (see table 6.4). Black stems correspond to the default configuration.

relatively low performance across all evaluation metrics. It has a low IDF_1 of 56.4 %, a low IDP of 70.4 % and a low IDR of 47.0 %, tracks only 22 trajectories for more than 80.0 % of their length and mostly loses 20 trajectories. It produces a high number of 5213 false negatives and has a low MOTA of 47.4 %. Only the achieved number of fragmentations (79) and the MOTP of 22.2 % are reasonable results. Another detector, that does not match the expectations, is the Faster R-CNN/NASNet model, which yields lower scores than the Faster R-CNN models based on Inception-ResNet v2 and ResNet-101 for almost all of the metrics, despite having the highest mAP of 43 and highest inference time among all detectors. Another metric failing short of the expectations is the frame rate, as it varies only very little among the tested detectors, despite the inference times being different by orders of magnitude. However, qualitatively the variations mostly meet the expectations. So, the frame rate for the Faster R-CNN/NASNet model with 53.19 Hz is lower than that of the detectors based on Inception-ResNet v2 (56.18 Hz), ResNet-101 (58.08 Hz) and ResNet-50 (58.40 Hz). Moreover, the low proposal Faster R-CNN/ResNet-101 as well as the Single Shot Detector achieve the highest frame rates of 61.59 Hz and 61.24 Hz, respectively. An exception is the R-FCN detector, which yields the lowest frame rate among all detectors (52.68 Hz).

6.4.2 Tracking Algorithm

As compared to the other parameters, the tracker used to track individual bounding boxes in intermediate frames shows only a small impact on most of the evaluation metrics. A slight increase in the MOTA from 63.4 % to 66.2 % is observable when switching from MOSSE over KCF to CSRT. However, this

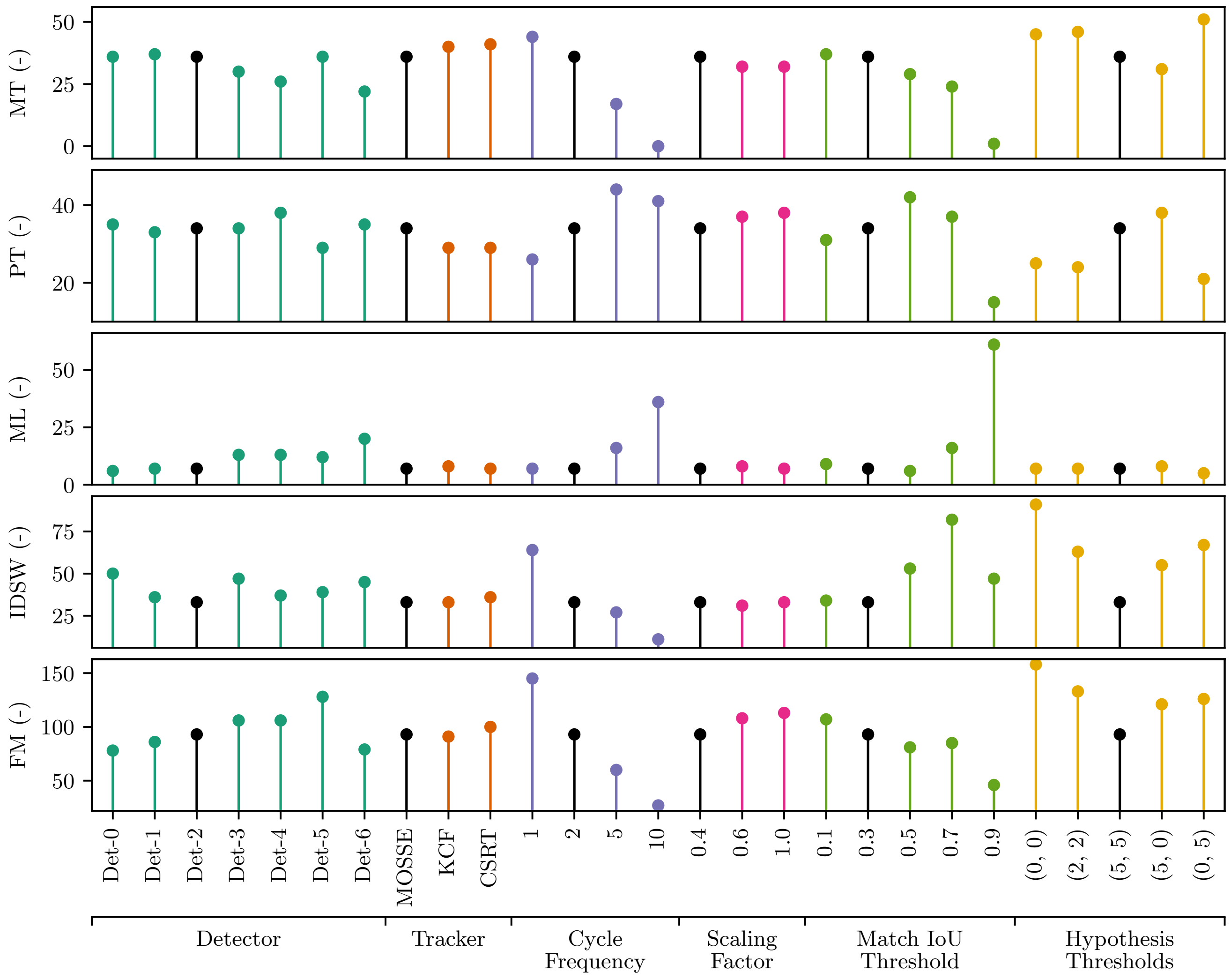


Fig. 6.7: Resulting track metrics (MT, PT, ML) as well as ID switch (IDSW) and fragmentation (FM) counts for all tested configurations (see table 6.4). Black stems correspond to the default configuration.

comes at the price of a decreased MOTP, which drops from 22.0 % to 20.6 %. CSRT provides slightly higher values for precision (90.1 %) and recall (74.8 %) as compared to MOSSE with a precision of 89.0 % and a recall of 72.7 % and generates less false positives and false negatives than the other two trackers. Furthermore, it correctly tracks 41 of the 77 ground-truth tracks for more than 80 % of the sequence length. But interestingly, CSRT generates 3 more ID switches and 9, respectively 7 more fragmentations than KCF or MOSSE. Again, it should be noted that these are subtle differences as compared to other experiments, such as changing the match IoU threshold. The only large impact can be noted with regard to the frame rate, which drops from 58.18 Hz for MOSSE over 10.40 Hz for KCF to only from 3.34 Hz for CSRT. This is notably the largest impact on the frame rate across all conducted experiments.

6.4.3 Detector Cycle Frequency

Changing the detection cycle frequency f_d from 1 over 2 and 5 to 10 shows an interesting trend in most of the metrics. Running the detector more frequently yields higher values for the IDF$_1$, IDR, recall, precision as well as MOTA and lower number of false positives and false negatives. It also leads to more ground-truth tracks being mostly tracked (MT). Thus, there seems to be a strong correlation between

Tab. 6.6: Official mAP and inference time of the utilized object detectors of the Tensorflow Object Detection library. [119]

Detector	**mAP**	**Time / ms**
FRCNN/NASNet (Det-0)	43	1833
FRCNN/Inception v2 (Det-1)	37	620
FRCNN/ResNet-101 (Det-2)	32	106
FRCNN/ResNet-101 (lp) (Det-3)	–	82
FRCNN/ResNet-50 (Det-4)	30	89
RFCN/ResNet-101 (Det-5)	30	92
SSD/ResNet-50 (Det-6)	35	76

these metrics and the detector cycle frequency. However, this is not given for the IDP metric, which varies, but without any correlation to the cycle frequency. The highest IDP of 81.6 % is achieved for $f_d = 2$. In case of the MOTP the inverse effect is observable as the metric decreases when the detector is run more frequently. Much stronger negative correlations with the cycle frequency seem to exist for the IDSW and FM metrics, which decrease from 64 ID switches and 145 fragmentations for $f_d = 1$ to only 11 ID switches and 27 fragmentations for $f_d = 10$. The frame rate is again positively correlated to the cycle frequency, yielding much higher frame rate values for less frequent operation of the detector (87.82 Hz for $f_d = 10$ compared to 8.83 Hz for $f_d = 1$). It has to be noted that the resulting metrics, such as IDF$_1$, IDR, MOTA, recall, precision and the number of false negatives, for detector cycle frequencies of $f_d = 5$ and $f_d = 10$ are among the worst across all analyzed configurations.

6.4.4 Frame Scaling Factor

The frame scaling factor, which determines the size of the input frame to the tracking system, shows a very interesting behavior as the highest scores for the IDF$_1$, IDP and IDR metrics as well as recall and MOTA are achieved for the lowest resolution of the input frame ($s = 0.4$). Additionally, this setting mostly tracks 36 of the 77 ground-truth trajectories, which exceed the other two tested scaling values by 4. For $s = 0.4$ also the lowest number of false negatives and fragmentations are achieved. However, the differences in the resulting metrics for the three tested values of the scaling factor are relatively small compared to the differences occurring when changing, for example, the detector cycle frequency or the match IoU threshold. Furthermore, there exist no apparent correlation between the scaling factor and the mentioned metrics, as the results for IDF$_1$, IDR, IDP, recall, precision, MOTA as well as the number of false positives and false negatives are the worst for a scaling factor of $s = 0.6$, and slightly increase again when the scaling factor is set to $s = 1.0$. In case of the precision, MOTP and number of false positives, the system achieves the best results for $s = 1.0$. The lowest number of 31 ID switches occurs at $s = 0.6$, while for $s = 0.4$ and $s = 0.6$ the number is 33. The only metric which shows a strong correlation with the scaling factor is the frame rate. Here, the frame rate decreases from 58.84 Hz for $s = 0.4$ over 37.30 Hz for $s = 0.6$ to 16.85 Hz for $s = 1.0$.

6.4.5 Match IoU Threshold

When increasing the value of the match IoU threshold θ_{IoU} from 0.1, over 0.3, 0.5 and 0.7 to 0.9, IDF$_1$, IDP, IDR, MOTA and recall decrease significantly, first marginally, then quickly to the lowest values of all experiments for $\theta_{\text{IoU}} = 0.9$. Here, the IDF$_1$ is only 13.7 %, the IDP 51.2 %, the IDR 7.9 %, the MOTA 13.6 % and the recall 14.7 %. These results are significantly lower than the best results for $\theta_{\text{IoU}} = 0.1$ with an IDF$_1$ of 76.0 %, an IDP of 83.6 %, an IDR of 66.7 %, a MOTA of 63.46 % and a recall of 74.3 %, where the IDF_1 and IDP are even the highest among all experiments. The number of mostly tracked (MT) and mostly lost (ML) ground-truth behaves very similarly and for $\theta_{\text{IoU}} = 0.1$ 37 of 77 trajectories are mostly tracked, while only 9 are mostly lost. For $\theta_{\text{IoU}} = 0.1$ the situation is inverted and only a single trajectory is mostly tracked, while 61 trajectories are mostly lost. The number of false positives stays at a relatively constant level of around 1100 for $\theta_{\text{IoU}} = 0.1$, 0.3 and 0.5 and then drops significantly to 673 and 86 for $\theta_{\text{IoU}} = 0.71$ and 0.9 respectively. Accordingly, the precision reaches its maximum of 94.5 % for $\theta_{\text{IoU}} = 0.9$, decreasing for smaller values of the IoU threshold. However, for values of 0.1, 0.3 and 0.5 the differences in the precision are small. The MOTP shows similar properties and stays relatively constant in the range of 22.0 % to 22.4 % for all match thresholds apart from $\theta_{\text{IoU}} = 0.9$, where it drops significantly to only 19.2 %. On the contrary, the number of false negatives increases only slightly from 3130 to 3690, when increasing the IoU threshold from 0.1 over 0.3 to 0.5, but then increases drastically to 4936 for $\theta_{\text{IoU}} = 0.7$ and 10397 for $\theta_{\text{IoU}} = 0.9$. This means for $\theta_{\text{IoU}} = 0.9$ the system misses over 85 % of all 12192 bounding boxes in the evaluation dataset. The number of fragmentations seems to correlate weakly with the match IoU threshold and linearly decreases from 107 fragmentations at $\theta_{\text{IoU}} = 0.1$ to 46 fragmentations at $\theta_{\text{IoU}} = 0.9$, whereby the value of 85 for $\theta_{\text{IoU}} = 0.7$ is slightly off the trend. The number of ID switches first shows an increasing trend rising from 33 for an IoU threshold of 0.3 to 82 switches for $\theta_{\text{IoU}} = 0.7$, but drops to a much lower value of 47 for $\theta_{\text{IoU}} = 0.9$. The frame rate also seems to be affected by the IoU threshold and decreases first slowly, then much faster from its initial value of 59.48 Hz for $\theta_{\text{IoU}} = 0.1$ to 42.77 Hz for $\theta_{\text{IoU}} = 0.9$. Changing the match IoU threshold also leads to an interesting change with regard to how bounding boxes are marked as valid. For large values of the threshold fast moving people are not marked as valid anymore and their bounding boxes stay in an invalid state.

6.4.6 Hypothesis Thresholds

Finally, a look at the impact of different values for the hypothesis validation and deletion thresholds θ_v and θ_d on the tracking performance provides further useful insights. Compared to other parameters, these two thresholds have no impact on the frame rate and only a small effect on the IDF_1, IDP, IDR, MOTA and MOTP metrics, although not as small as for the different tested tracking algorithms and scaling factors. However, the impact on all other metrics is relatively large. In case of the three value pairs $(0, 0)$, $(2, 2)$ and $(5, 5)$ for θ_v and θ_d, the IDF_1, IDP and IDR increase slightly. For the thresholds $(5, 0)$, the IDF_1 and IDR decrease, while the IDP increases. For the thresholds $(5, 0)$ it is the other way round and the IDF_1 and IDR increase, whereas the IDP drops significantly. The MOTA is at a relatively high level of 69.5 % and 69.0 % for the thresholds $(0, 0)$ and $(2, 2)$ respectively, and by 5.0 % to 7.4 % lower for the three other analyzed value pairs. The MOTP seems unaffected by the choice of the threshold parameter apart from the parameter values $(0, 5)$, which lead to an increase in the MOTP of 0.7 %. The recall decreases slightly for increased values of the two thresholds from 79.0 % for the thresholds $(0, 0)$ to 66.9 % for the thresholds $(5, 0)$, however it increases to 82.0 % for the thresholds $(0, 5)$, which is the highest recall value of all 27 experiments. The precision stays relatively constant for the thresholds $(0, 0)$, $(2, 2)$ and $(5, 5)$, but reaches a much higher value of 93.9 % for thresholds $(5, 0)$ and drops to a very low value of 82.4 % for thresholds $(0, 5)$. Accordingly, the number of false positives is at a constant level of around 1068 for the first three threshold pairs, while it drops to 528 for thresholds $(5, 0)$ and increases to 2130 for thresholds $(0, 5)$, which is the highest value observed in all experiments. The number of false negatives shows a similar behavior as the IDP metric, increasing for the first four tested threshold pairs and dropping again for the last pair $(0, 5)$. The numbers of false negatives achieved for the thresholds $(0, 0)$, $(2, 2)$ and $(0, 5)$ are 2563, 2675 and 2195 respectively, and are the lowest of all experiments. Looking at the numbers of mostly tracked, partly tracked and mostly lost ground-truth trajectories reveals that the highest scoring configuration is $(0, 5)$, which mostly tracks 51 out of all 77 ground-truth trajectories and loses only 5 trajectories, which are the highest values for MT and ML of 27 tested configurations. The MT drops to only 31, increasing the PT to 38, while maintaining a low ML of 8 for the thresholds $(5, 0)$.

In terms of ID switches and fragmentations the qualitative behavior is equal, yielding high numbers of 91 and 63 ID switches as well as 158 and 133 fragmentations for the thresholds $(0, 0)$ and $(2, 2)$, respectively. For thresholds $(5, 5)$ the lowest number ID switches (33) and fragmentations (93) are achieved, however they increase for thresholds $(5, 0)$ and $(0, 5)$, with IDSWs of 55 and 67 and FMs of 121 and 126. Thus, setting the thresholds to values others than $(5, 5)$ leads to numbers of ID switches and fragmentations, which are higher than in almost any other experiment.

6.5 Qualitative Results

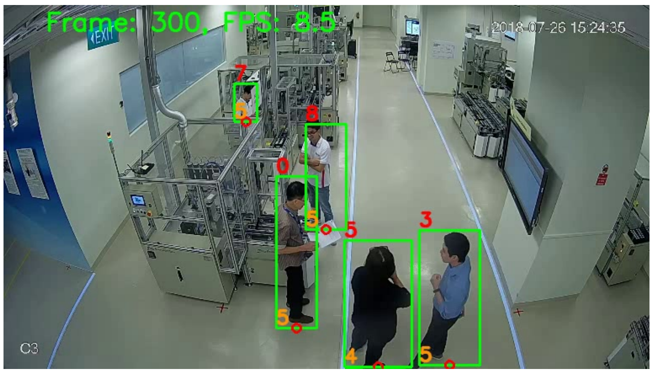
The system performance can not only be analyzed in terms of evaluation metrics, but also qualitatively by looking at i) the frame-wise predictions for bounding boxes B and box IDs I and ii) the generated output trajectories in world coordinates (X, Y). Analyzing the output data qualitatively helps to built a better understanding of the system and reveals potential flaws which are not captured by the evaluation metrics.

Some frame-wise system outputs were already shown in fig. 5.3 and fig. 5.5, whereby the latter illustrated the principle of the low-pass filter. Further output frames with predictions are depicted in fig. 6.8. This collection contains four common scenarios observed in the output video of the detection and tracking system. The first one is the most frequent case in which all targets are correctly tracked and marked as valid. The second group consists of frames containing at least one false positive, which is correctly marked as invalid, and thus filtered out. The third common scenario are people entering the scene. As shown, it takes a while until the match counter of the target has reached the hypothesis valid threshold and is marked as valid. This leads to a delayed recognition of new targets. Not depicted, but similarly, it takes a while before a target is deleted when leaving the scene. The fourth group of common events shows a similar issue. Here, targets are marked as invalid, when moving too quickly.

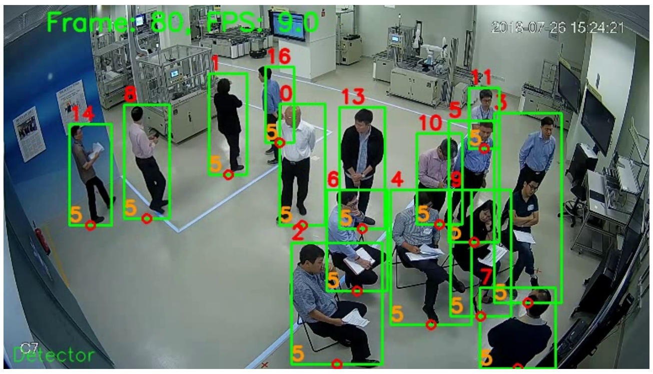
As mentioned before, the Shopfloor Monitor stores target trajectories in a database. Visualizing these trajectories allows for a quick validation of the correct function of the Shopfloor Monitor. As an example fig. 6.9 shows the trajectories extracted from the five evaluation videos. The given trajectories can also be derived with respect to the time, yielding the targets' velocities. Figure 6.10 shows the distribution of velocity magnitudes of all targets for each of the evaluation sequences. The output trajectories can be further analyzed, which is used to improve the tracking performance, and support tasks, such as activity recognition and person identification.

6.6 Conclusion

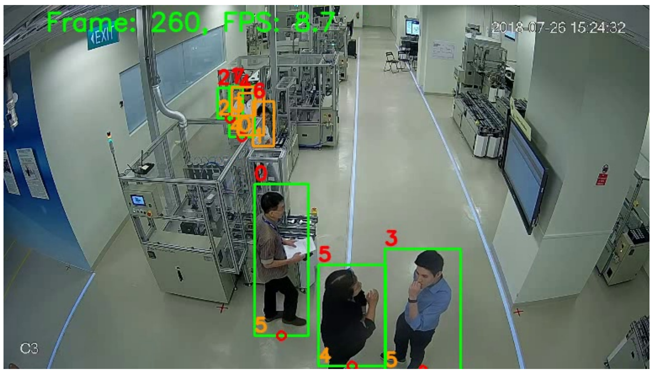
In this chapter a thorough experimental analysis of the detection and tracking subsystem of the Shopfloor Monitor was performed. First, an application-specific evaluation dataset containing five video sequences of a total length of 888 frames with over 12000 manually annotated bounding boxes was created to ensure applicability of the evaluation results to the given use-case. The next step was a literature review on evaluation metrics. Suitable metrics were identified and described in detail. Afterwards, the experimental design of the study was developed and a tool for automatic execution and reporting of the experiments was implemented. The experimental results were computed and then visualized and described in detail. Additionally, qualitative outputs of the detection and tracking system were shown to gain a better understanding of the practical performance. In the the next chapter the experimental results are discussed in detail.



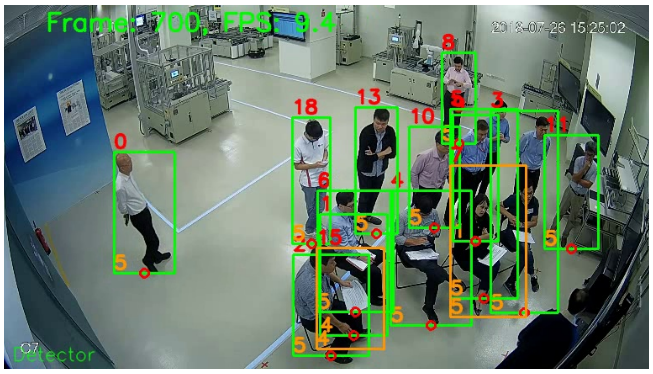
(a) All targets valid (C_3, $n = 300$).



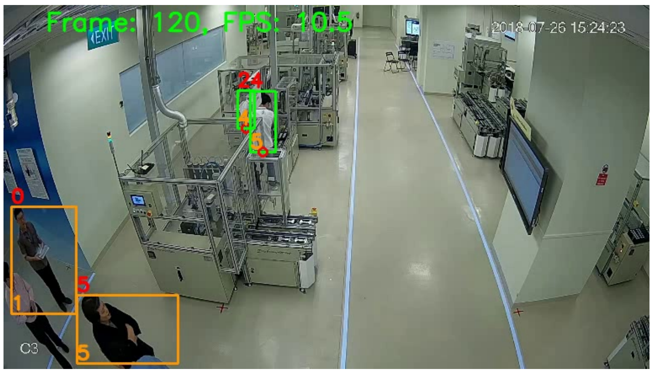
(b) All targets valid (C_7, $n = 80$).



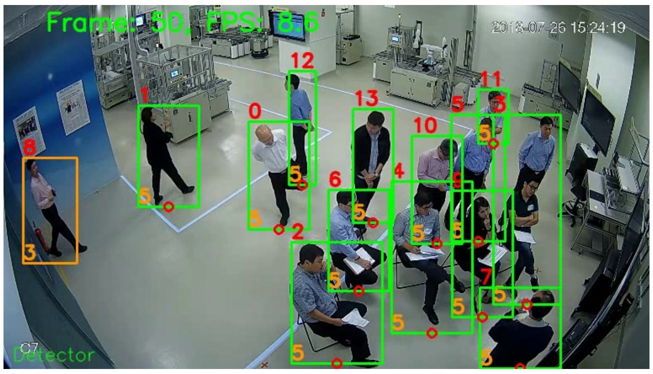
(c) Some false positives correctly filtered out (C_3, $n = 260$).



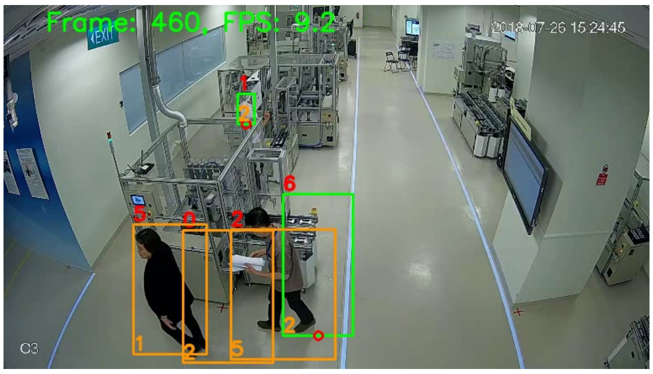
(d) Some false positives correctly filtered out (C_7, $n = 700$).



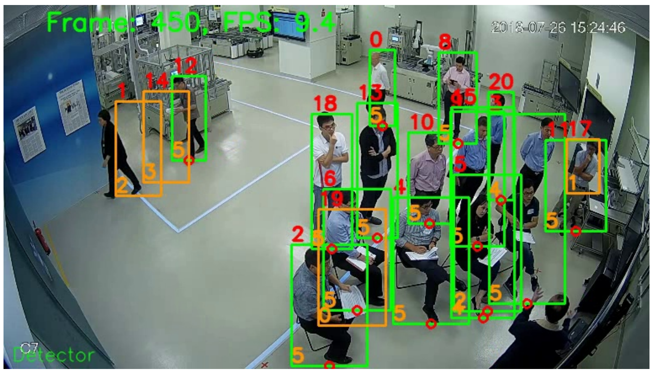
(e) Invalid targets due to entering (C_3, $n = 120$).



(f) Invalid targets due to entering (C_7, $n = 50$).

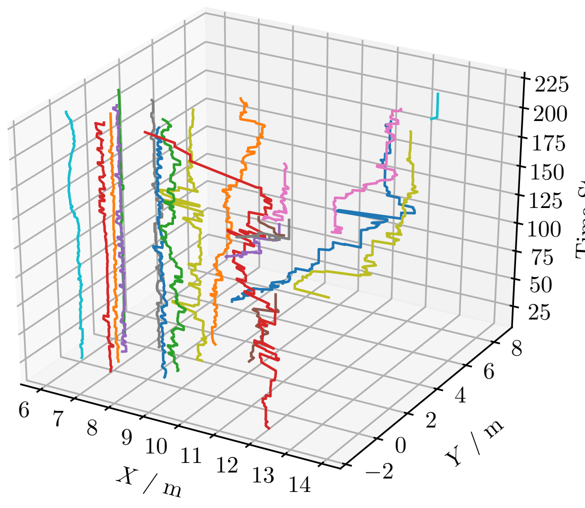


(g) Invalid targets due to fast movements (C_3, $n = 460$).

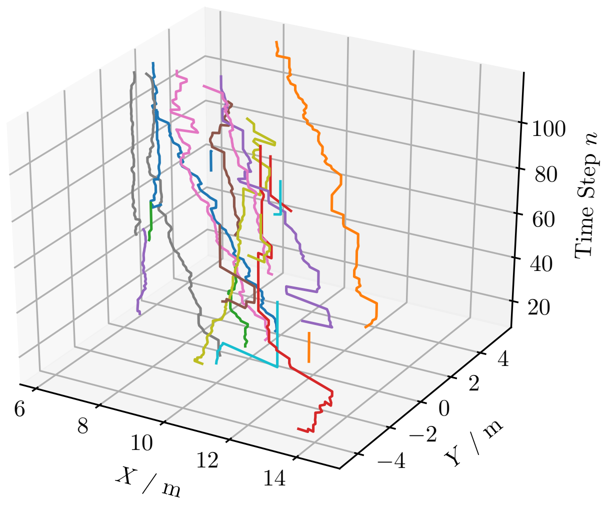


(h) Invalid targets due to fast movements (C_7, $n = 450$).

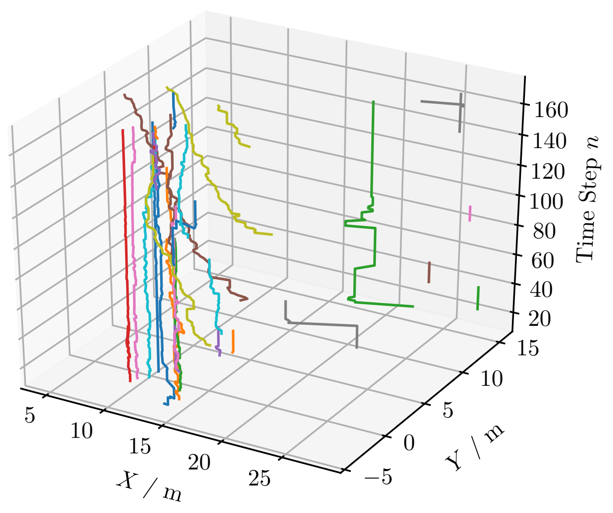
Fig. 6.8: Qualitative results of the detection and tracking framework. The left and right column contain frames of camera C_3 and C_7, respectively. The frame number is given as n. In the first row, all targets are correctly tracked and marked as valid. The second row contains some false positives, which are correctly marked as invalid (orange box). In the third row, targets are marked as invalid because they have entered the scene and the match counter has not reached the hypothesis valid threshold yet. The last row illustrates an issue with fast moving targets which are incorrectly labeled as invalid.



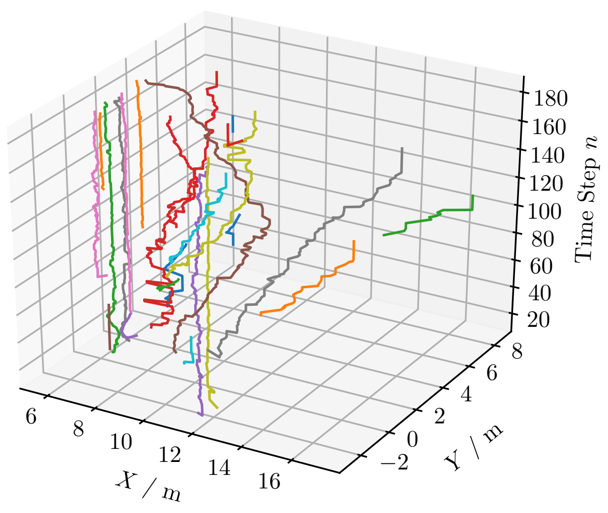
(a) Target trajectories in Seq-0.



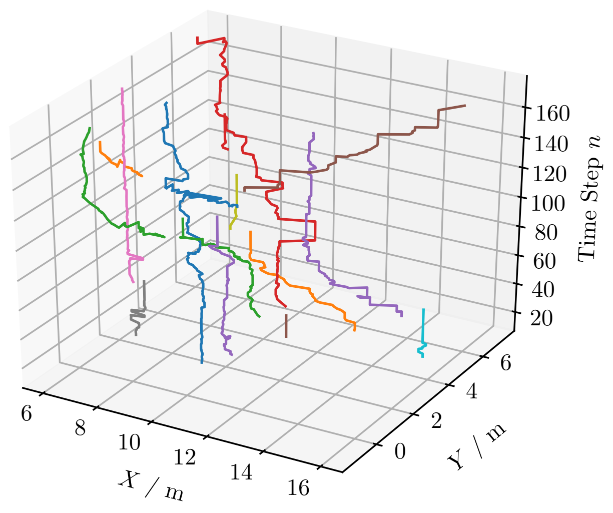
(b) Target trajectories in Seq-1.



(c) Target trajectories in Seq-2.

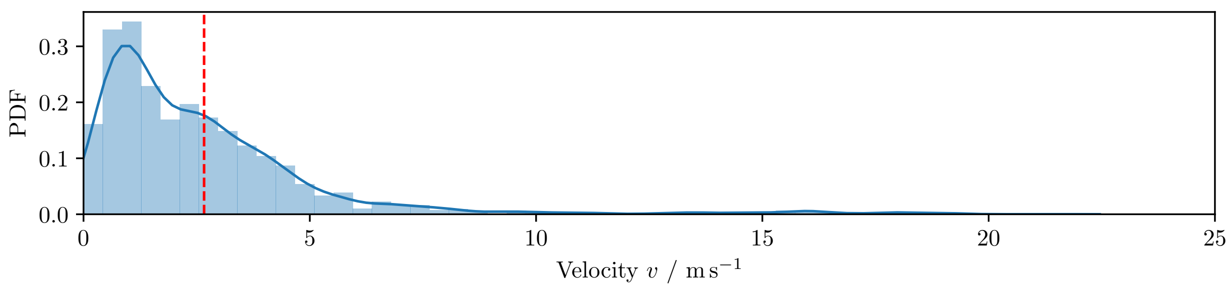


(d) Target trajectories in Seq-3.

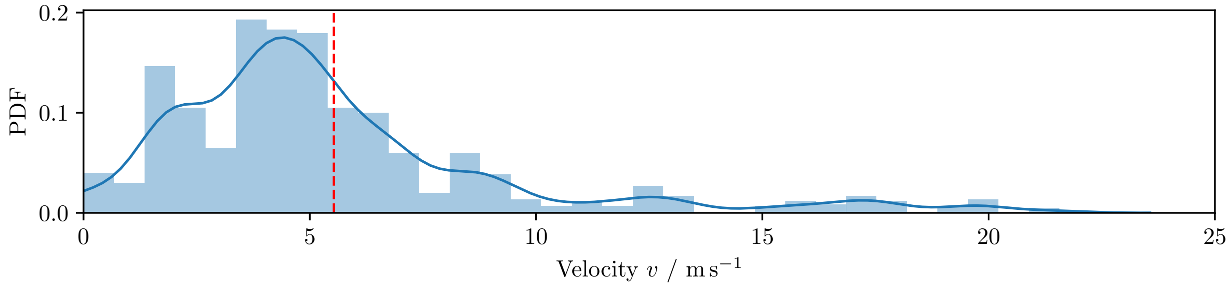


(e) Target trajectories in Seq-4.

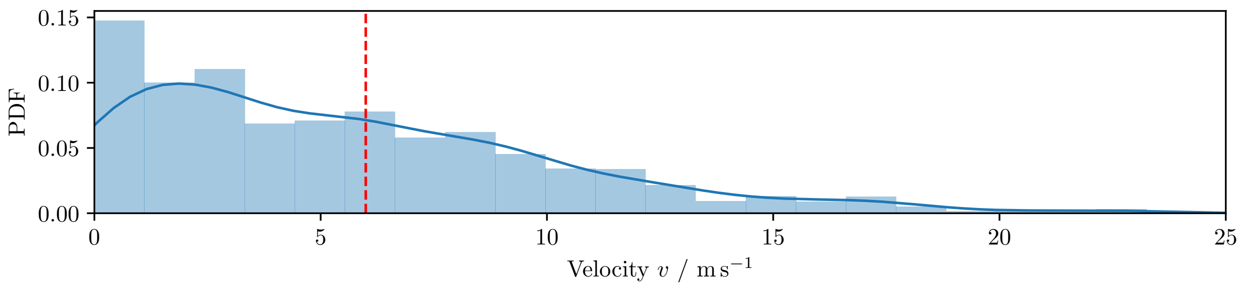
Fig. 6.9: Target trajectories acquired by the Shopfloor Monitor. World positions (X, Y) on the horizontal axis are plot versus discrete time steps n on the vertical axis. Each subplot corresponds to one of the five evaluation sequences. Different colors indicate different persons.



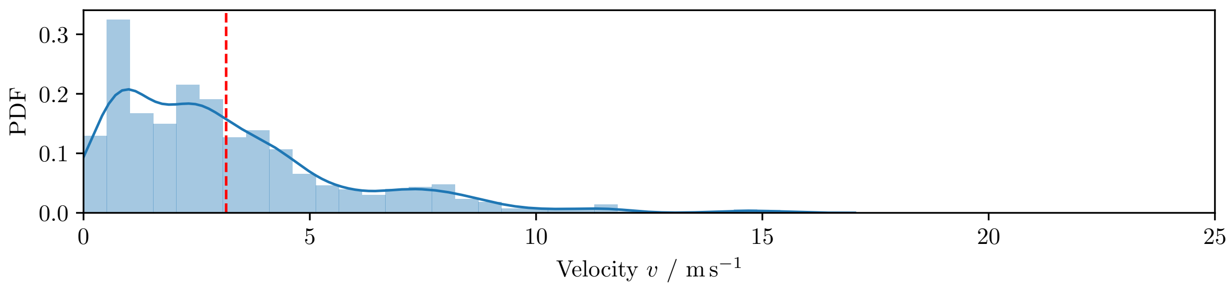
(a) Velocity Distribution of targets in Seq-0.



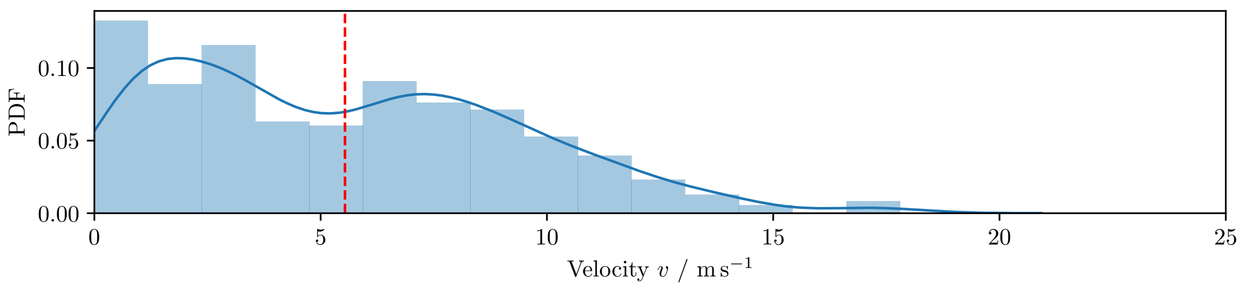
(b) Velocity Distribution of targets in Seq-1.



(c) Velocity Distribution of targets in Seq-2.



(d) Velocity Distribution of targets in Seq-3.



(e) Velocity Distribution of targets in Seq-4.

Fig. 6.10: Velocity distributions of all targets on the five evaluation sequences. Each subplot corresponds to one of the five evaluation sequences. Shown are univariate kernel density estimates. Velocities are computed as $v = dS/dt$ with $dS^2 = (X_{n+1} - X_n)^2 + (Y_{n+1} - Y_n)^2$ and $dt = t_{n+1} - t_n$. The mean of each distribution is shown as a red dashed line.

Chapter 7

Discussion

This chapter discusses the experimental results presented in chapter 6. The results are interpreted in section 7.1 to build a profound understanding of the characteristics and potential problems of the developed detection and tracking system. The importance of each individual system component for the overall tracking performance and processing speed is examined in section 7.2, and unsuitable parameter values are identified in section 7.3. Afterwards, section 7.4 proposes three sets of hyperparameters which yield either a faster or a more accurate detection and tracking system. Finally, the experiments are critically reflected in section 7.5. This includes the utilized evaluation metrics and the created evaluation dataset. Possible issues are highlighted and improvements are suggested.

7.1 Interpretation of the Results

This section interprets the analysis results presented in section 6.4 and follows the same structure. It discusses the analysis results of different object detectors, tracking algorithms, detector cycle frequencies, frame scaling factors, match IoU thresholds and hypothesis thresholds. Finally, the qualitative results shown in section 6.5 are discussed.

7.1.1 Detection Algorithm

Interpreting the analysis results for the different object detectors is a challenging task because of the black-box nature of the underlying deep neural networks, and the collected results are far from sufficient to provide detailed explanations for most of the observations. However, detectors can be roughly classified into high-, medium- and low-performing models. So, the Faster R-CNN models with NASNet, Inception-ResNet v2 and ResNet-101 base networks perform best on most of the evaluation metrics, apart from the frame rate and the MOTP. The high performance of these three models is facilitated by the superiority of the Faster R-CNN meta-architecture in combination with the very large base networks. Using the smaller ResNet-50 base network performance in most of the metrics immediately drops. Only the frame rate increases because a smaller base networks means a lower amount of computations and thus shorter inference time. A similar explanation holds for the Faster R-CNN/ResNet-101 model with a reduced number of bounding box proposals. Here, the speed-up is caused by the lower amount of bounding boxes, which have to be classified and regressed. This comes at the cost of an increase in the number of false negatives and a drop in the precision because some targets might be missed. Accordingly, the number of ID switches and fragmentations increases, while the MOTA and number of mostly tracked trajectories drop. The comparably lower performance of the R-FCN/ResNet-101 model, especially the high number of generated false positives and track fragmentations, can be explained by the fundamentally different meta-architecture. The lowest performing object detector, with respect to most of the metrics, is the SSD/ResNet-50 model, due to the small base network and also to the meta-architecture, which has been designed with the goal of a high processing speed, at the expense of a decreased accuracy. Despite being qualitatively correct, the measured frame rates for the different object detectors require special attention because the quantitative differences are smaller than indicated by the official metrics in table 6.6. The reason for this is that the object detector is run only on every second frame, which largely eradicates the differences in the processing times of the detector models.

7.1.2 Tracking Algorithm

The analysis results show that all metrics, apart from the frame rate, appear mostly unaffected by the tracking algorithm used. This seems surprising at the first glance, considering the fact that MOSSE and KCF are 8, respectively 2 years older than CSRT, and official metrics suggest that CSRT outperforms the other two trackers by a large margin [13]. However, the qualitative trend of CSRT being slightly better than KCF, which itself marginally improves on the results of MOSSE, is indeed recognizable. The reason for the small differences is that for all three tests the default parameter for the detection cycle frequency is set to $f_d = 2$, which means that the detector (Faster R-CNN/ResNet-101) is run on every second frame and resets the tracker states accordingly. Thus, there is not enough time for tracking errors to accumulate and the superiority of CSRT over KCF and MOSSE does not appear. The fact that the processing speed of the tracker is not affected by how often its state is reset, explains the large drop in the frame rate when switching from MOSSE over KCF to CSRT.

7.1.3 Detector Cycle Frequency

The results for different values of the detector cycle frequency f_d seem mostly plausible. The less frequently the object detector is run, the more ground-truth targets are missed, which is mirrored by the increasing number of false negatives and the drop in the recall. Accordingly, the decreasing IDF$_1$, IDR and MOTA metrics for increasing f_d can be explained. This also accounts for the decreasing number of mostly tracked (MT) ground-truth trajectories and increasing number of partly tracked (PT) and mostly lost (ML) trajectories. The string decrease in ID switches and fragmentations can be explained by the fact that the number of misses is so large that less than half of the targets get detected at all, and if the number of detections is very low, so is the number of ID switches and fragmentations. The change of the number of false positives with different cycle frequencies is less intuitive as the number first increases with increasing cycle frequency f_d, but then drops again for $f_d = 10$. An explanation for this might be the accumulation of tracking errors for $f_d = 10$, leading to less frequent redetections and thus a smaller number of valid tracking targets (indicated by the MT metric) and less false positives. However, for $f_d = 1$ to $f_d = 5$ the accumulated tracking errors are small enough to lead to a high number of valid targets and consequently more false positives. The precision directly correlates with the number of false positives and true positives. Thus, the decreasing and for $f_d > 5$ stagnating trend of the precision is understandable. Equally comprehensible is the strong correlation between the detector cycle frequency and the frame rate as the inference time of the object detector makes up a significant amount of the total processing time of a single frame. Thus, running the detector less frequently increases the average frame rate. The IDP drops slightly for cycle frequencies other than $f_d = 2$, however does not show any systematic correlation with the cycle frequency. Finding an explanation for this is as challenging as accounting for the slightly increasing MOTP for less frequent runs of the detector. Further experiments are needed to find the causes of this behavior.

7.1.4 Frame Scaling Factor

As mentioned in the results section, altering the frame scaling factor yields very counter-intuitive results, whereby the highest scores for IDF$_1$, IDP, IDR, MOTA, precision and recall are achieved for the smallest frame size with scaling $s = 0.4$, being slightly lower than the results for the original frame size. The worst scores occur for the intermediate frame size with scaling factor $s = 0.6$. The same qualitative behavior recurred for the number of false positives and false negatives as well as the MT, PT and ML scores. This behavior can not be explained on the basis of the current experimental results. Further experiments with different scaling factors are needed to validate the trend or reveal the experiment for $s = 0.6$ as outlier. Only a single metric, such as the number of false positives or false negatives, might show a strange behavior and thus affect all other dependent metrics. Another reason could be a higher accuracy of the detector on a smaller input frame. Despite the qualitatively strange behavior, it has to be noted that the differences in the metrics due to varying the scaling factor are relatively small as compared to varying other parameters. Thus, the scaling factor plays only a subordinate role for the tracking performance. This does not apply to the frame rate, though, which depends significantly on the scaling factor because tracking of targets in a larger image requires substantially more calculations and thus more time. Moreover, increasing the scaling factor and hence the image size positively affects the MOTP, which is comprehensible, because a larger image with a higher resolution allows for more precise tracking of targets, and thus reduces localization errors.

7.1.5 Match IoU Threshold

As stated earlier, the match IoU threshold θ_{IoU} strongly correlates with most of the evaluation metrics and shows a mostly consistent behavior. The higher the match IoU threshold, the larger the overlap between a detected and a tracked bounding box has to be during an update step of the tracker to produce a successful match. Thus, an increase of this threshold leads to a decrease in matches and accordingly an increase in false negatives. The number of false positives drops with the threshold since there are less valid tracking targets for higher IoU thresholds. As precision and recall directly correlate with the number of true positives, false positives and false negatives, the resulting trends are intelligible. The same applies for the MT, PT and ML metrics, as well as the IDF$_1$, IDP, IDR and MOTA. The decrease in the number of fragmentations with an increasing θ_{IoU} is apparently linked to the decreasing number of successful matches, which leads to a low number of generated trajectories and consequently a low probability for fragmentations of those trajectories. A similar explanation can be given for the number of ID switches, which initially increases and then collapses for $\theta_{\text{IoU}} = 0.9$ due to the low number of true positives that could experience an ID switch. As this number is higher for the other tested values of θ_{IoU}, the amount of ID switches is also larger, growing with an increasing θ_{IoU} due to the higher rate of misses and redetections of targets leading to the assignment of a new ID. The drop in the, otherwise constant, MOTP for $\theta_{\text{IoU}} = 0.9$ is less obvious, but might be reasoned by a computational inaccuracy due to the low number of true positives, which contribute to the calculation of the average IoU distance (see eq. (6.4)). The slight decrease of the frame rate for increasing values of θ_{IoU} indicates an increase in the computational load. This is likely to be an implementation issue, causing the tracking system to use more computing ressources in case of a low number of valid bounding boxes.

7.1.6 Hypothesis Thresholds

The hypothesis valid and deletion thresholds θ_v and θ_d act as a low-pass filter on the bounding boxes, preventing them to erroneously pop into existence or disappear with every false positive or false negative produced by the object detector. Thus, a large impact of these two thresholds on the performance metrics is expected. This holds true for the number of ID switches and fragmentations of trajectories which decrease for increasing values of θ_v and θ_d as bounding boxes and their IDs are temporally more stable. Setting one of the thresholds back to zero disables the filters for either false positive or false negative detections, and thus increases the number of fragmentations and ID switches. The impact of θ_v and θ_d on other metrics, such as IDF_1, IDP and IDR, is qualitatively as expected, although not as large as initially assumed. This might be due to the fact that the detector is only run on every second frame, thus reducing the impact of erroneous detections on the tracking result by itself and calling for the low pass filter needed less urgently. The numbers of false positives and false negatives produced by the combined detection and tracking system can be explained well with the filter analogy because the higher θ_v and θ_d are, the more detections are filtered out and marked as false negatives. Keeping $\theta_v = 5$ and setting $\theta_d = 0$ increases the number of false negatives even further as boxes are deleted immediately after being missed by the detector. The next five detections of this target are then also missed because it becomes valid only after five consecutive redetections. The number of false positives is low in this case because trackers are immediately deleted after the first miss and thus can not produce any false positives. In case of a higher θ_d, the system has to produce multiple false positives to decrease the deletion counter before the target is deleted. This is especially true for the case $\theta_d = 5$ and $\theta_v = 0$, where bounding boxes are spawned immediately, even for false positives of the detector, leading to many invalid bounding boxes not assigned to any target, which accumulate many false positives during their decay. Here, the number of false negatives is very low because bounding boxes are immediately created, missing only very little detected targets. With these explanations the results for precision and recall as well as the MT, PT and ML metrics can be justified, as those scores directly correlate with the numbers of false negatives and positives. The same holds true for the MOTA, even though it is less obvious because the MOTA is computed from not only the false negatives and false positives, but also the number of ID switches. The decrease of the frame rate with an increase of θ_v and θ_d coincides with the previous assumption about an increasing computational load for a larger number of false negatives. As the change in the number of false negatives is only little, the frame rate also changes only slightly. If the implementation exhibited a constant resource-utilization, the frame rate should not be affected at all by changing the hypothesis thresholds.

7.1.7 Qualitative Results

This section discusses the qualitative results described in the first part of section 6.5. As shown in fig. 6.8, targets entering or leaving the scene are not recognized or deleted immediately, but with a certain delay. The delay when entering is caused by the fact that the match counter has to reach the hypothesis valid threshold θ_v first. This means, the detector has to correctly detect the target θ_v times before it is marked as valid. Similarly, when a target leaves the scene, its associated match counter has to decrease to the hypothesis deletion threshold θ_d before it is deleted. Thus, the low-pass filter introduces a delay. This delay can be minimized either by running the detector more frequently (smaller f_d) or by reducing the hypothesis valid and deletion thresholds. However, running the detector more frequently reduces the processing speed, and reducing the hypothesis thresholds causes more false positives and misses. Thus, a sensible choice for θ_v, θ_d and f_d has to be made to achieve a reasonable performance. These parameters also directly impact the tracking performance on fast moving targets. If the detector is run less frequently (larger f_d) and the hypothesis valid and deletion threshold are small, fast moving targets are more likely to be incorrectly missed and marked as invalid. This is what happens when the tracker is unable to follow a fast moving target. Thus, the tracked bounding box and detected bounding box have only a small IoU and the target is not correctly redetected. Consequently, the match counter is decreased until it reaches the hypothesis deletion threshold θ_d. In this case, the target is marked as invalid and becomes valid again only upon slowing down its motion. Again, this problem can be reduced by choosing different settings for θ_v, θ_d and f_d. However, these changes will again increase the number of false positives and misses.

A solution to this problem would be adaptive scheduling of the detector as presented in [180]. Instead of running the detector in fixed intervals, a scheduler network observes the tracking quality and decides to run the detector if the tracking quality decreases. This causes the detector to run more frequently if a target moves fast and less frequently if all targets move slowly. Similarly, the delay when entering or leaving the scene is reduced.

7.2 Parameter Importance

To wrap up the discussion, system parameters are classified by the extent of their impact on the overall tracking performance and processing speed (see table 7.1). This helps to decide which parameters of the detection and tracking system have to be adjusted to change the operating point and achieve a certain behavior. As can be seen from the table, all parameters have a large impact on either speed or tracking performance or even both. This underlines the importance of all hyerparameters and the modules controlled by those, and validates the design of the developed detection and tracking system. The parameters with major impact on the processing speed are the detector cycle frequency, the tracking algorithm and the frame scaling factor, whereby adjusting the latter two influences the other performance metrics only little. Thus, to improve the processing frame rate of the system, these three parameters should be optimized first. The object detector also has an impact on the processing speed, which, however, strongly correlates with the detector cycle frequency. Thus, running the detector more frequently, increases its impact on the overall processing speed. The match IoU threshold and hypothesis thresholds have hardly any effect on the frame rate and, hence are not suitable for speed-optimization. But their impact on the tracking performance is comparatively large, which also holds true for the object detector and the cycle frequency. It is interesting that only the detector and the cycle frequency affect both tracking performance and speed simultaneously, whereas the other four parameters have a decoupled impact on tracking performance and speed. This allows for accurately adjusting the operating point of the monitoring system.

Tab. 7.1: Impact of the analyzed system parameters on overall tracking performance and processing speed. Tracking performance is used as a synonym for all performance metrics apart from the frame rate. A large impact of the respective parameter is marked $+$, while a medium and small impact are indicated by $\circ$ and $-$, respectively.

Parameter	**Performance Impact**	**Speed Impact**
Detector	$+$	$\circ$
Tracker	$-$	$+$
Cycle Frequency	$+$	$+$
Scaling Factor	$-$	$+$
Match IoU Threshold	$+$	$-$
Hypothesis Thresholds	$+$	$-$

7.3 Invalid Parameters

In addition to analyzing the qualitative influence of different parameters, the study also revealed limits and disadvantageous values for some of the parameters, which lead to very low performance scores or an insufficient processing speed. Most remarkable is the match IoU threshold $\theta_{\text{IoU}} = 0.9$, and to some extent also $\theta_{\text{IoU}} = 0.7$, which results in a heavily diminished tracking performance. The same applies for the detector cycle frequencies $f_d = 5$ and $f_d = 10$. The value $f_d = 1$, on the other end of the spectrum, leads to an inadequate processing speed, and the high numbers of fragmentations and ID switches render this configuration useless. Other, in terms of processing speed, invalid configurations are the scaling factors

$s = 0.6$ and $s = 1.0$ as well as the use of KCF- or CSRT-trackers. The latter could be solved by running trackers for individual targets in separate processes. Further configurations showing low or among the different metrics less consistent scores are the ones that use R-FCN and SSD as detectors. The outstanding numbers of ID switches and fragmentations of the hypothesis threshold $(0, 0)$ illustrate the importance of using a low-pass filter on target-detections. Not using the filter for delaying the acceptance of proposed target-detections ($\theta_v = 0$ and $\theta_d = 5$) is not a solution given the high number of false positives.

7.4 Useful Parameter Configurations

Based on the analysis results and the knowledge gained in this section, three system configurations with different emphasis on tracking performance and processing speed are selected. As most of the tested values for the cycle frequency, scaling factor, match IoU threshold and validation thresholds have turned out to produce unfavorable results, leaving only one valid value, all three configurations will use a detector cycle frequency $f_d = 2$, a scaling factor $s = 0.4$, a match IoU threshold $\theta_{\text{IoU}} = 0.3$ and hypothesis thresholds $\theta_v = 5$ and $\theta_d = 5$. Using KCF- or CSRT-trackers leads to a disqualifyingly low frame rate due to the sequential updating of all trackers. Thus, all three configurations use MOSSE as a tracker, leaving only the object detector as means to adjust the detection and tracking system towards either a higher processing speed or higher tracking performance. The first configuration uses Faster R-CNN/ResNet-101, which provides a balance of speed and tracking performance. Replacing the detector with Faster R-CNN/Inception-ResNet v2 in the second configuration slightly increases the tracking performance at a small cost of processing speed. The third configuration, finally, uses the Faster R-CNN/ResNet-50 detector, rendering it the configuration with lowest tracking performance but highest processing frame rate.

7.5 Critical Review of the Experiments

After having discussed the analysis results, the conducted parameter study itself has to be critically examined with special focus on the design of experiments, the utilized evaluation metrics and the created dataset. A minor issue of the design is the low number of 27 analyzed parameter combinations, which is justified by the need for a low running time ($\approx 1\,\text{h}$) of the evaluation tool to quickly assess the impact of new features on the tracking performance. To improve this, the number of experiments and thus analyzed parameter configurations could be increased, and a smaller, mutable subset of these experiments could be selected when running the evaluation tool. This would provide a deeper insight into the behavior of individual parameters. The generated data could be used to build a model of this behavior, which could then assist in the search for better performing and more subtly balanced parameter configurations than the ones presented above.

Allowing for more parameters would also facilitate a more fine-grained analysis of the utilized tracking and detection algorithms, which have their own sets of hyperparameters. For example, the implementations of KCF and CSRT in OpenCV have 17 and 26 tunable hyperparameters, respectively, and the object detection models provide configuration files for adjusting the number of box proposals, non-maximum supression, dimension of the input image and layer parameters, such as strides, kernel sizes and regularization. However, in this study none of those parameters was modified and only the default configurations were analyzed, thus withholding valuable insights which could assist in improving the tracking system.

Another aspect, which could be improved, is the quantitative analysis of parameter correlations, which would yield an understanding of how parameters affect each other, and thus disclose possibly redundant parameters, which could be combined, replaced by a simpler parameter or decoupled by a new set of parameters to gain more control of system behavior.

As shown in section 6.2, there are additional metrics for assessing tracking performance. Computing those would provide further insights. Especially, relative metrics, such as false alarm rate, relative number of ID switches or relative number of fragmentations, which compute events with respect to other metrics, the number of frames or number of total occurrences, might prove useful. Furthermore, running the tracker at different operating points by varying the distance (IoU) threshold for target matching across frames during the metric computation, would enable plotting of performance curves, such as a precision-recall-curve, overlap-sucess-curve or precision-distance-curve. Profiling the components of the tracker software in terms of memory and time complexity and measuring utilization of input and output ressources, time latency as well as size of internal buffers, would provide additional metrics, which could be especially helpful for optimizing the implementation of the tracking system in future. However, as these values have a statistical nature, similar to the frame rate, multiple runs of the evaluation tool would be necessary and thus conflict with the goal of a low computing time of the analysis.

Other concerns about the conducted analysis relate to the evaluation dataset, which might not cover the entire bandwidth of expectable events. For example, there is no scene with people running or falling to the floor included. Moreover, the case of switched off main lights and activated infrared night vision system of the cameras is covered just as little as scenes which are dedicated to other target categories than humans. Hence, if the monitoring system is expected to see the mentioned scenarios, an extension of the evaluation dataset with according sequences might be required to ensure significance of the provided results. It might also be useful to compute the evaluation metrics on the official MOT dataset, which covers a broader range of activities, although not being as specific to the given task of static camera surveillance in a manufacturing environment. This would also allow for comparing results with those of other tracking systems proposed by the research community.

Certainly, the biggest issue of the conducted analysis is the single-view character of the evaluation dataset and utilized MOT metrics. As the Shopfloor Monitor comprises multiple cameras with possibly overlapping fields of views, additional challenges, such as handover between viewports and multiple detections of the same target in different camera images, have to be considered when evaluating the tracking system. Moreover, special multi-camera evaluation metrics and a multi-target multi-camera evaluation dataset, like DukeMTMC [161, 181, 182], should be used.

Apart from these flaws, the conducted study meets the initial requirement of investigating the impact of the system components and parameter values on tracking performance and processing speed. Especially, the automated evaluation script provides a valuable tool for the fast analysis of introduced features and automated reporting of performance in the future development process of the monitoring system. Despite showing the minor deficiencies mentioned earlier, the created evaluation dataset is a solid basis for examination of the system performance under application-oriented conditions. The large discrepancies of the performance metrics of the object detectors computed on this dataset versus the ones computed on the COCO dataset emphasize the importance of a suitable evaluation dataset for obtaining of meaningful results. Finally, the study results could be utilized to propose three system setups, each providing a different speed-accuracy trade-off and thus being useful for a specific application.

7.6 Conclusion

In this chapter the results of the experiments conducted in chapter 6 were exhaustively discussed. The relations between parameter values and resulting tracking performance and processing speed was highlighted. These findings were used to propose three different system configurations with different performance characteristics. Finally, the experiments were critically investigated and possible problems as well as improvement suggestions were emphasized. In the following chapter possible applications of the Shopfloor Monitor are described.

Chapter 8

Applications

This chapter introduces possible applications for the developed Shopfloor Monitor. Only applications in the domain of manufacturing and production technology are considered. Some of the applications, such as occupancy detection or the indoor positioning system can also be transferred to other domains. Implementing the proposed ideas in a manufacturing facility, might significantly increase the productivity and decrease production cost. Not all proposed applications are new, but they built on existing ideas and alter them or introduce novel aspects. It is important to note that some of the applications require the Shopfloor Monitor to be extended with additional algorithms, such as activity recognition or face identification.

8.1 Occupancy Detection

An obvious application is occupancy detection, the task of sensing the presence of people in rooms or specific areas in a manufacturing facility. Occupancy detection can be realized with different sensors, such as microwave, ultrasonic, infrared or power meters. Cameras as sensors enable more detailed insights, as the produce spatially and temporally rich data, which is human-understandable. Cameras can easily detect when people enter or leave a room or a building. By counting these events, the number of people in a room, a building or a specific area can be estimated. With full coverage of the site, this number can be retrieved directly from the video footage and yields a more accurate estimate. Occupancy detection is an established practice and widely used in modern buildings to control lighting systems [183] as well as heating, ventilation and air conditioning (HVAC) [184]. This increases energy efficiency of buildings and manufacturing facilities. In some applications, such as dry-rooms for battery production or clean-rooms for micro-electronics, controlling HVAC based on the occupancy might benefit the manufacturing process. Recent surveys on occupancy detection are [185] and [186].

8.2 Machine Waiting Queue

A more specific use case is the detection of resource or machine utilization via installed surveillance cameras. If the number of machines for a process step is limited, workers might have to queue to use a machine. To prevent idling of workers, they can register at a counter. The installed cameras can then be used to detect, when the limited resource is vacant again, and notify the next registered worker in the waiting queue. While such a situation shows a major flaw in the production process, and is thus unlikely to occur in larger manufacturing facilities, it might still arise in smaller companies, research institutes or laboratories with limited resources.

8.3 Staff Activity Recognition

The system under development can also be used to classify activities of human workers, such as standing, walking, running, talking, lying or working. Recent surveys on video-based activity recognition are [187], [188] and [189]. Staff activity data provides useful insights into the efficiency of individual workers or the entire team. It can be used to determine the times during a day, in which a team is most distracted and most concentrated. This knowledge aids in the design of more efficient and less exhausting manual production processes. It might also be used to identify major sources of distraction in production sites. Moreover, automatically recognizing abnormal [190], inappropriate or potentially dangerous behavior of staff prevents accidents. In case of large warehouses, staff dishonesty and theft lead to high annual losses [191–193], which can easily be prevented by camera-based activity recognition. Being able to classify actions of human workers also aids in further applications, such as emergency assistance and risk assessment, which are covered in the next sections. Monitoring staff activity comes along with ethical issues, which have to be considered, when designing a related application.

8.4 Emergency Assistance

A virtual building information model (BIM) can assist in emergency situations, such as a fire. It provides rescue forces with information about the building layout, hazardous materials and location of extinguishing devices [194, 195]. When combined with the proposed system, the virtual BIM can be enriched with real-time information about the location and the current condition of people in a building. This enables more efficient rescue operations, as people in a more critical condition can be evacuated first. Additionally, the BIM can be extended with real-time information about the location and type of hazardous materials across the building or manufacturing site. Furthermore, location and size of fire sources, propagation direction of the fire, and strength and direction of smoke emission can be detected, and embedded into the BIM. This aids path planning in rescue operations and planning of efficient extinguishing strategies. It can also be used to estimate the best escape route, and suggest it to occupants via LED lighting on the floor or loudspeaker announcements. Information about the fire can also be used to directly trigger extinguishing systems, such as sprinklers or flooding systems. To enable such functionality, the camera system has to withstand the conditions during the fire. It might also be necessary to extend the system with sensors, which are not affected by smoke, for example radar sensors. Apart from fire emergencies, the camera system can also aid in detecting smaller accidents throughout the manufacturing facility, such as collisions between vehicles and workers or injury of staff. The type and severity of the accident, as well as potential injuries, can be automatically assessed, and appropriate rescue actions can be initiated.

8.5 Automatic Risk Assessment

Another application is automatic risk assessment in manufacturing environments. Risk assessment is the task of identifying hazards, determining potential consequences and their likelihoods, deciding about the tolerability of a risk, and reporting the findings [196]. When using real-time video footage of installed surveillance cameras in combination with human and object tracking as well as activity recognition, these task could easily be automated. For example, such as system can identify, whether personnel wears the required protective clothing, when working on machines or in hazardous environments. It can automatically notify the worker about the risk. Similarly, potentially dangerous behavior of staff in hazardous zones of a production site can be detected. As the camera system can see the entire production site, it can estimate future trajectories of vehicles and people, and thus can be used to foresee and prevent collisions by automatically stopping a vehicle or warning the person. Besides collisions, a wide range of other accidents might be anticipated and prevented by fast computer vision algorithms. When storing hazardous materials, surveillance cameras can be used in combination with analysis algorithms to permanently monitor and predict the condition of storage containers. This can prevent accidents due to erosion or defect of the containers. During handling or processing of dangerous goods, the proposed system can detect spill or leakage and inform the staff. The camera system can be used in conjunction with human pose estimation to monitor, whether people work under ergonomic and healthy conditions. It can also be used to monitor working and break times of personnel and advice to take a break in regular intervals. Automatic assessment of risks enables thorough and continuous logging of hazardous events in a database. Analyzing this data can help to make the manufacturing site more ergonomic and safer.

8.6 Smart Access Control

Another application of the camera system is to monitor and control access to specific areas in a manufacturing facility. In the simplest case, this means granting or prohibit access through a gate or door. But it can also mean restricting access to specific locations, which are otherwise not protected against unauthorized access. For example a hazardous machine tool or piece of equipment can be surrounded by a virtual fence. The camera system can detect, when a person passes through this fence and initiate a security alert, notify a foreman or stop the machine. In combination with face recognition [197–202], people can be identified, and individual access rights for each person can be granted. This enables people with higher access rights, for example a production engineer, to move freely across the site, whereas people with lower access rights, such as machine operators and visitors, are prohibited access to most of the zones to their own safety. The same principle can also be used to control, which personnel has access to which machines. This prevents misuse of equipment or use of machines by unauthorized or unqualified staff. The same system can also serve as a simple intrusion detection system, which is activated outside of the working hours. Access restriction can also serve the purpose of protecting a sensitive manufacturing process, for example in micro-production or in the production of pharmaceutical goods.

8.7 Control of Autonomous Ground Vehicles

Autonomous ground vehicles (AGV) are widely used for transportation of materials and products across manufacturing sites [203–206]. Some of these vehicles follow fixed and specifically marked paths via path tracking [207, 208]. Others use integrated sensors, such as cameras, Lidar or ultrasonic sensors, for simultaneous localization and mapping (SLAM) [209–213]. Further challenges, which have to be solved to enable efficient operation of an AGV, are obstacle detection, collision avoidance, scheduling and path planing [214–216]. While most of the approaches use on-board sensors to reduce complexity of the system and installation cost, external cameras can be used to provide the individual vehicles with additional information about the global state of the vehicle fleet. This includes location and operating status of other vehicles as well as detection and avoidance of congested routes. The system can aid in computing a globally optimal path for all vehicles in a fleet, and enables fast rescheduling of operation in case one or multiple vehicles in a fleet fail. A centralized system, which knows about the state of every vehicle in a fleet, also enables to solve tasks in a cooperative manner to save time and maximize utilization of all vehicles.

8.8 Safety System for Industrial Robots

Being indispensable in modern manufacturing environments, the number of operational industrial robots worldwide in 2017 was 2.098 million, with numbers estimated to grow up to 3.788 million in 2021 [217]. During the past years, the robotics landscape changed from strictly separating human workers and industrial robots with protective fences to shared work spaces and direct collaboration. During fence-less robot operation the physical safety fence is replaced with a virtual safety fence. Cameras observe the area around the industrial robot and slow it down or stop it, whenever a human enters the hazardous area [218–221]. Human Robot Collaboration (HRC) [222–224] goes a step further and enables direct collaboration between a human worker and the robot. HRC depends on advanced methods in computer vision and motion tracking to estimate relative motion between human and worker and initiate according reactions [224]. Given the recent advances in computer vision, the proposed camera system can act as a sufficient data-source for either fence-less robot operation or HRC.

8.9 Indoor Positioning System

Another potential use case is an Indoor Positioning System (IPS), which aids people in finding their location and navigating within GPS denied buildings or factory halls [225–228]. Installed surveillance cameras can be used in combination with people detection and tracking to locate a person in the factory. The location can be sent to a handheld device, such as a smart phone, smart watch or smart glasses, and visualized on a map. This requires identification of the person, for example via facial recognition. As an alternative to displaying the location on a map, external indicators, such as LED lights on the floor or walls, can help a person to navigate through a manufacturing facility. A destination can be selected on one of multiple tablets or simple keypads, which are distributed across the factory. The system can also automatically recognize, whether someone is a visitor. It can guide him or her to the next registration counter or to the person, he or she has an appointment with. Similarly, the system might instruct external delivery workers about the desired placement of delivered material or goods within the factory. A navigation system with optical indicators might also assist workers in large warehouses, when storing, removing or moving goods.

8.10 Product and Resource Tracing

Tracing materials, products and equipment throughout the entire production process is an established method in modern factories [4]. It aids the optimal scheduling of production orders, automatic reporting of the production process and obviates tedious, time-consuming and inaccurate manual logging [229]. One approach to tracing is to equip products and machines with microcontroller-based tags, which store product and order related information and log information about each conducted process step [230, 231]. Communication with other products, machines or analysis terminals is mostly realized via Radio-Frequency Identification (RFID), Wi-Fi, Bluetooth or GSM [229, 232, 233]. Instead of equipping every single product with a smart tag, the proposed camera system can be used, in conjunction with object detection and identification algorithms, to trace resources and materials throughout the production process. The system can detect, which materials or products enter a machine. This information can then be logged in a central database along with additional data, retrieved directly from the machine. Omitting smart tags, and centralizing the tracing system, keeps its complexity low and thus reduces development, installation and maintenance cost. A centralized camera-based system is not restricted to fixed terminals for reading out product data, but can locate products anywhere on the production site. The system might also be used to detect potential loss, wasting or theft of materials and products.

Furthermore, localization and identification of products and materials enables to compare the current state of the entire manufacturing facility with a desired state. From the observed deviations potential problems and solutions can be inferred. For example, a worker might leave a pallet with products at the machine, instead of transporting it to the warehouse, as intended by the production schedule. The system can detect this discrepancy and notify the worker or a foreman. This approach can also be used to automatically examine, whether the production site is left in the desired state after working hours.

8.11 Automatic Quality Control

Computer vision-based quality control is used throughout many production domains [234, 235], such as the food [236–238], textile [239, 240], semiconductor [241] and automotive industry [242], assembly [243] and additive manufacturing [244]. Other applications are in the production of ceramics [245], the steel industry [246] and in manual manufacturing processes [247]. The system developed in this work can be used for the automatic quality control of products. It can also verify the order and correctness of process steps. Omitted or erroneously executed process steps can be detected and reported. This is especially helpful in manufacturing or assembly processes with a high number of manual tasks, for example in the assembly of automotive or aerospace products. But also in fully automated production systems, the camera system provides an additional, independent layer of process monitoring. For example, it can detect breakdown or malfunction of machine tools, robots or transport systems. As the camera system overviews the entire production plant, it can provide valuable information about a problem, which helps to resolve it without without affecting production.

8.12 Automatic Reporting

As mentioned, the camera system can automatically generate reports for the risk assessment or the product and resource tracing. Of course, it is not limited to this, but can report arbitrary events. This replaces time-consuming, error-prone and among human workers unpopular, manual reporting tasks. Modern analytics tools based on machine learning [248] can be leveraged to generate additional insights from the raw data, which might have remained unnoticed with traditional, manual reporting. Using the camera system speeds up the reporting procedure, frees workers and makes them available for more important tasks. Moreover, generated reports are more concise, thorough, constant in style and immediately digitized. Automated reporting allows to capture a higher number of events, which can provide valuable information in case of accidents or malfunctions. An example of automated reporting is the assignment of a worker to a process step, obviating signing or stamping of an according document unnecessary. Another example is the automated documentation of experiments in research laboratories. Used materials and equipment, process steps and measurement results can be automatically gathered and combined in a report.

8.13 Process and Factory Analysis

Collecting large amounts data within a factory, leverages a wide range of analysis and control applications, which help to improve production processes, resource planning and plant monitoring. Ultimately, this leads to *smart factories* with more efficient, faster and more flexible manufacturing processes, reduction of waste as well as cheaper and more individualized products [2–4, 249]. Currently, video data is rarely used for more than monitoring and quality control of individual processes (see section 8.11), however could supply existing plant information systems, such as Operations Management Systems (OMS) and

Resource Planning Systems (RPS) [250], with spatially and temporally rich information about the ongoing production processes.

For example, video data can be harnessed to analyze and improve the factory layout. Despite being planned and simulated thoroughly before installation, a plant layout might still have problems, which were not considered during planning. Examples are the suboptimal placement of tools, machines or storage areas, unnecessary long walking distances for human workers, congested transport routes, bottlenecks in the material flow and overflowing storage areas. Problems like these are not unlikely, because the optimal plant layout is always a tradeoff between multiple competing target quantities, which minimizes a specific, manually designed, cost-function. Moreover, this cost function is a simplified and incomplete model of the real-world and the underlying optimization procedures of the Facility Layout Planning (FLP) are NP-hard and often ill-posed [251–256]. Video data, showing the flow of resources, materials and products through the manufacturing plant, provides a helpful tool for the practical evaluation of a factory layout. It might uncover unused potential and weaknesses of an existing factory layout. In case of modular learning factories [257, 258], which comprise of several interconnected and freely arrangeable modules, cameras can be used to detect the current factory layout and feed this information back into the production control system. The same cameras can be used to analyze activities and attentiveness of students working on the learning factory.

Video data can also be used to analyse production processes. For example, it might be used to analyze different scheduling strategies in terms of machine utilization, productivity and cost. Furthermore, various modifications of the production processes can be analyzed. Such modifications are the transition of a production line towards a new product, the replacement of an old machine, an extension of a production line with additional machines, factory layout changes, the implementation of a new manufacturing technology, a change of the personnel or fluctuation of the markets. All this data can also be used for process control, rescheduling in case of plant malfunctions and for automated planning of new production processes [259].

8.14 Virtual Factory Model

Virtual manufacturing [260–262] and the virtual factory [263–268] are recent topics in manufacturing technology. Here, a virtual model of the entire factory, including all processes, is created. It is used to simulate and monitor processes and aids the management in decision-making. It enables evaluation of different process parameters and factory layouts without affecting the actual production. The virtual model can also be used for simple walk-throughs [269], to subjectively analyze the factory layout. Moreover, it can be used to simulate the production of a new product prior to altering the production line. Existing research already focuses on incorporating live process data into the virtual model [270, 271], however video data was not considered yet. This is interesting, because video data provides a practically feasible and cost-effective method to incorporate human workers and arbitrary objects into the virtual model. Inclusion of humans and objects is crucial to accurately model the manufacturing process and predict future events, such as accidents, plant malfunctions or the next maintenance operation. Applications of such a virtual factory model, containing both process data and live information about human workers and arbitrary objects on the shopfloor, are manifold. For example, the model can assist supervisors in monitoring plant related data and overseeing a team of human workers. As the virtual factory model is a compact representation of many video streams, a single supervisor might be sufficient to monitor a large number of human workers. Supervisors can also use the model to assign tasks to individual workers on the shopfloor. Anomaly detection [272–274] can guide the attention of the supervisor to interesting events. Other applications of the virtual factory model are emergency simulations [275] and training of human workers in a virtual setting prior to deployment into the actual work place [276–279].

A life-updated virtual model enables virtual collaboration and communication between people, located at different physical locations. This includes teams, which monitor and optimize the factory layout or production processes [280–282]. But it also refers to individual workers on the shopfloor, who can use the virtual model to follow the locations and activities of their colleagues. This gives them an overview of the current situation. Additionally, the model can provide workers with details about the current task of their collaborators, helping them to collaborate more effectively with their team colleagues. This is especially important in large and complex scenarios, where colleagues have no direct contact to each other. The model can also be used to communicate with team colleagues. For example, a production manager, located off-site, could easily meet with engineers and operators in a virtual meeting room, which is part of the factory model.

8.15 Virtual Product Model

An obvious extension of the virtual factory model is a live-updated model of the product. This is especially helpful in cases of large and complex products, which take several days or even weeks to finish, such as in the aerospace or shipbuilding industry. Cameras can be used to monitor conducted process steps and assess the current status of the product. This helps workers, engineers and managers to keep an overview of the already conducted tasks and to decide upon the next steps. By comparing the conducted steps against a model of the desired process, failures can be spotted and corrected. Video data also enables to measure certain aspects of the product. For example during assembly of an airplane, the dimensions and relative position of components can be measured and compared against the desired values. Similarly, vibration modes of the airplane could be measured during vibration analysis of the structure, superseding the need for cost intensive manual installation of vibration sensors [283, 284]. Such measurement data can then be assigned to the virtual product model for later analysis and quality control. People can virtually walk through the product model and view data and documents related to sub-components at the according location of this component. For example during assembly of an aircraft, a person can walk into the cockpit of the aircraft and click on an installed component, retrieving all related documents, such as the technical datasheet, details about the manufacturer and the manufacturing process, licensing documents and reports of the quality control.

8.16 Conclusion

This chapter introduced possible applications of the developed Shopfloor Monitor in the domain of manufacturing technology. They aimed at improving safety, productivity and flexibility of the manufacturing environment, reduce cost and downtime, secure machines and assets, open up new ways of team-work and communication, and improve process control and plant simulations by incorporating the human factor. The next chapter discusses privacy concerns of the developed system.

Chapter 9

Privacy Concerns

The use of video cameras in the Shopfloor Monitor for the permanent monitoring of human workers in a factory is accompanied by various ethical and privacy issues. This chapter briefly introduces possible solutions for rendering the system compliant with prevailing regulations.

To reduce the intrusiveness of the Shopfloor Monitor and protect the employees of a company, rules should be established and their compliance strictly controlled. For example, the system should not be used to monitor and report the performance of workers. Such a privacy violation could increase dissatisfaction and stress levels of the workers and would eventually lead to a decrease in performance, affect health and increase the number of absentees. Thus, the system should solely be used to assist the employees by taking on some of their tasks. This increases productivity of the workers, but at the same time reduces their work loads.

Most countries have already established laws to protect the privacy of employees and regulate the use of video surveillance and data collection. To comply with these regulations, the Shopfloor Monitor needs to be modified. For example, instead of storing the acquired video footage, only extracted high-level meta data, such as trajectories, should be stored. The original footage should be deleted immediately. To make this approach more secure, an IT security concept should be developed and implemented to prevent attackers from stealing or modifying data. Using smart cameras, which directly incorporate detection, tracking and mapping functions and only return meta data instead of the original video footage is another way towards a more secure system. To further protect the privacy of staff, the system should not record any audio. It should also allow for marking zones, such as private offices, as *private* and prevent collection of meta data, as soon as people enter these zones. Automatic identification of humans in surveillance videos might not be allowed in some countries. In these cases, the system must not implement appearance, face or gait recognition or any other means of identifying people. The possibility of identifying personnel based on the meta data only, should also be avoided. However, this might not always be easy to realize and restricts the functionality of the system. Some countries, for example Germany, require to unambiguously inform every employee about the camera monitoring and the further utilization of the acquired data. This can be done via signs, notice boards and written notification. Every employee has to be given the chance to opt-out of the monitoring. Such an employee could be recognized via visual markers on the clothing, which tell the system not to track this person. If compliance with the regulations can not be reached despite all these modifications, different, less intrusive sensors than cameras can be used. Examples are ultrasonic sensors, passive infrared sensors (PIR), radar, thermal cameras, magnetic sensors or pressure mats [185, 285]. However, these sensors provide less information-rich output data and might severely reduce the functional scope of the monitoring system. In cases, where camera monitoring of humans is completely prohibited, the Shopfloor Monitor can still be applied, but should focus on objects instead of humans. Humans could be detected by the camera hardware and disguised in the output footage, for example by masking or blurring. Use cases for such an object-centric monitoring system are the observation of material flows, automatic layout detection of flexible factories and control of transport vehicles.

If some or all of these modification are considered, the Shopfloor Monitor can assist employees, take over unpopular or tedious tasks, and reduce their stress levels without restricting their freedom and privacy.

Chapter 10

Conclusion

The research question of this thesis was how one can implement a system which acquires and processes human-related data in manufacturing environments in real-time using only a network of multiple surveillance cameras. Motivation for this issue was the weak utilization of human-related data in modern factories. As human workers still play an important role in modern manufacturing companies, they should be considered in the digitalization of the factory. Human-related data can be used for process control, analysis and simulation. Furthermore, it enables new applications, which boost productivity, flexibility and safety of the production, reduce cost and downtime and open up new possibilities for team-work and communication. Thus, monitoring humans helps to improve production and business processes, and allows a company to keep up with or outpace competitors.

With the background of this research issue, the overall goal of this thesis was specified. A novel system, named the *Shopfloor Monitor*, for the acquisition of human-related data in manufacturing facilities should be developed and experimentally evaluated. This system should only use a network of multiple surveillance cameras to localize human workers on the shopfloor. To demonstrate usefulness of the generated data, an example web app for real-time visualization of the tracking results within a virtual 2D floorplan of the factory had to be developed.

Prior to implementing the Shopfloor Monitor system, the requirements were analyzed and a specification was derived in chapter 2. In chapter 3 fundamentals of object detection, object tracking and multi-camera multi-target tracking were presented. These topics formed the theoretical foundation for the later implementation of the Shopfloor Monitor. Subsequently, a literature review was conducted in chapter 4. Existing solutions for multi-camera multi-target tracking and visualization were surveyed and their limitations revealed. It turned out that all solutions used outdated detection and tracking algorithms to localize humans within the video streams. Furthermore, no solution dedicated to applications in manufacturing technology could be found. This gap served as a further motivation for the present research issue. In chapter 5 the implementation details of the Shopfloor Monitor were covered. First, a modular, extensible and scalable software architecture was designed. At the core of this architecture was the detection and tracking module which localized humans in multiple parallel video streams. To push the state of the art recent detection and tracking algorithms were utilized. The deep learning-based object detector periodically detected humans in video frames and initialized or updated the trackers belonging to the targets. To match detections and trackers, the Hungarian method was used. A low-pass filter helped to reduce the impact of false positive and false negative detections, and stabilized the tracking. Camera calibration provided a method to transform locations of people from image to world coordinates. This allowed to plot trajectories of human workers in the visualization web app, which was implemented as part of a general, extensible app framework. The acquired trajectory data was stored in a database for usage in future web apps. In the end of chapter 5, limitations and possible future improvements of the current implementation, including the camera calibration, were discussed. In chapter 6 experiments were conducted to evaluate the implemented system. First, an evaluation dataset was created. It contained 5 video sequences with a total length of 888 frames and in total 12192 manually annotated ground-truth targets. Commonly used evaluation metrics, such as MOTA, MOTP, precision, recall and ID-based metrics were selected as basis for the evaluation and explained in detail. Afterwards, an experimental strategy was designed. This strategy aimed at exploring the impact of different sub-components of the detection and tracking framework on the overall tracking accuracy and processing speed. Apart from using 7 different object detection algorithms and 3 different trackers, a range of values for 4 further hyperparameters of the detection and tracking framework was analyzed. A script for automated execution of the experiments and logging of the results was developed. Finally, resulting evaluation metrics were presented. The results were discussed thoroughly in chapter 7. Additionally, the conducted study was critically reviewed. Limitations were revealed and future improvements suggested. In chapter 8 an overview of possible applications in the domain of manufacturing technology was given and, finally, ethical and privacy concerns of the developed monitoring system were examined in chapter 9 and possible modifications were presented to comply with existing regulations.

The following list summarizes the main contributions of this work:

- Setup and calibration of a multi-camera system within a factory.
- Development of a multi-camera multi-target detection and tracking framework based on state-of-the-art detection and tracking algorithms.
- Design of a modular and scalable software architecture for data acquisition, processing and long-term storage.
- Implementation of an extensible web framework and an example web app for real-time visualization of tracking results.
- Experimental evaluation of the detection and tracking framework based on MOT metrics and an application-specific dataset.

Comparing these contributions with the goals of this thesis specified in the introduction, it can be stated that this work obtained a satisfactory solution for the issue of a multi-camera multi-target tracking and visualization system. Figure 10.1 shows the developed solution in operation.

Despite improving the state of the art of multi-camera multi-target tracking and visualization solutions by using recent detection and tracking algorithms, the presented Shopfloor Monitor still lacks some features which are present in existing solutions. For example, handovers of targets between adjacent camera views are not managed, and redundant target detection by multiple cameras are not merged. Furthermore, target appearance is not considered during the matching of detections and trackers. Implementing these features would be necessary to perform better than state-of-the-art solutions. However, the decision was made to exclude these features and instead focus on the utilization of recent detection and tracking algorithms. Implementing the mentioned features is infeasible for a single developer and does not add any scientific value.

The conducted work opens up a multitude of future research directions. For example, the Shopfloor Monitor could be improved by incorporating the mentioned features into the detection and tracking framework. Alternatively, the developed multi-stage detection and tracking framework could be replaced by a unified deep learning model which takes multiple video streams as inputs and outputs trajectories in world coordinates. Such a model provides a large scientific value, as it learns to implicitly perform multi-camera multi-target tracking. This model could even include additional steps, such as coordinate transformation and trajectory post-processing. Future work could also focus on deploying the Shopfloor Monitor in one or multiple factories and quantitatively evaluate the impact of the system on productivity, downtime and cost. Another research direction is the development of further web applications to realize more of the use cases presented in chapter 2. Future work could also find additional use cases or apply the developed approach to other domains, such as monitoring in private homes or urban monitoring.

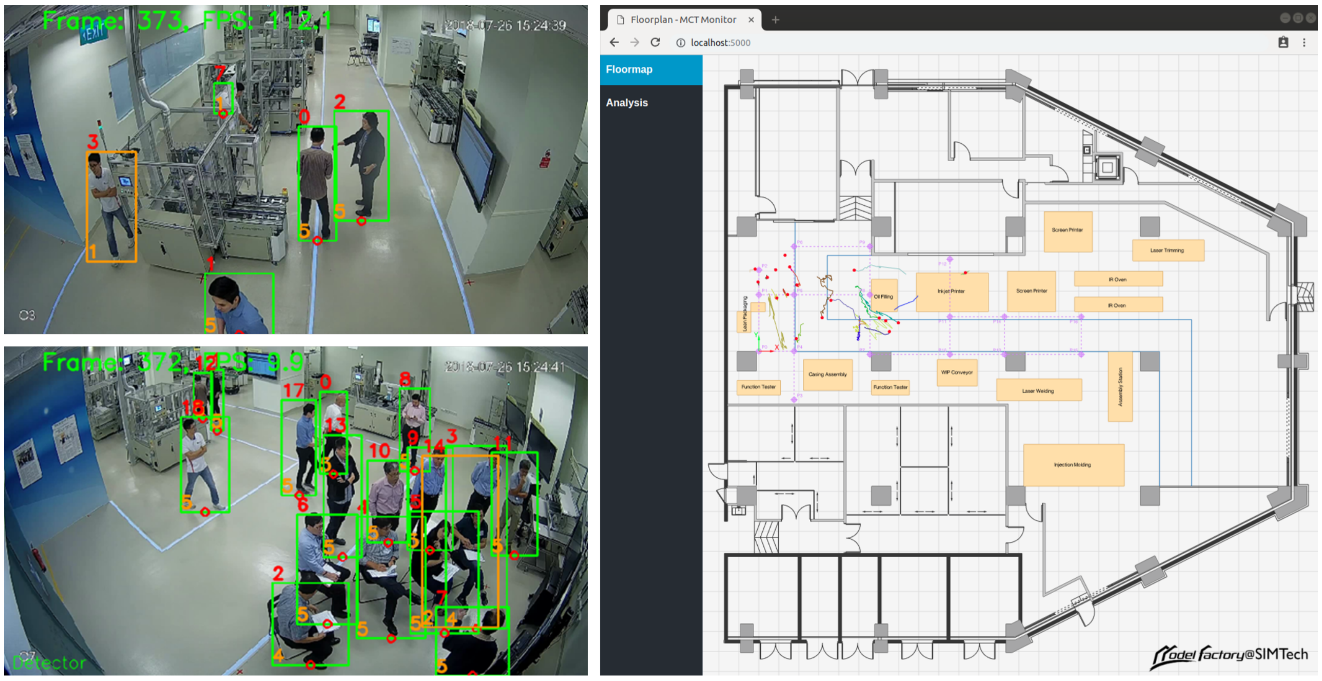


Fig. 10.1: Screenshot of the developed Shopfloor Monitor, showing raw video streams with tracked targets (green boxes) and the visualization web app running in a web browser.

The research pursued in this thesis is highly relevant for the future of manufacturing technology. As long as factories are not fully automated, human workers are a key factor in production processes. Thus, it is crucial for the success of future manufacturing enterprises to monitor human workers and utilize the collected data to simulate, control and optimize manufacturing and business processes. The presented work is the first step towards a more efficient, safer and resource-saving manufacturing landscape of tomorrow.

Bibliography

[1] *Declaration of Authorship.* 2017. URL: https://www.ent.wi.tum.de/fileadmin/w00bcx/www/Ehrenwoertliche_Erklaerung_deutsch_und_englisch.pdf (visited on 2018-08-17).

[2] Kang, H. S., Lee, J. Y., Choi, S., et al. “Smart manufacturing: Past research, present findings, and future directions”. In: *International Journal of Precision Engineering and Manufacturing-Green Technology* 3.1 (2016), pp. 111–128. ISSN: 2288-6206. DOI: 10.1007/s40684-016-0015-5.

[3] Wang, S., Wan, J., Di Li, et al. “Implementing Smart Factory of Industrie 4.0: An Outlook”. In: *International Journal of Distributed Sensor Networks* 12.1 (2016), p. 3159805. ISSN: 1550-1477. DOI: 10.1155/2016/3159805.

[4] Brettel, M., Friederichsen, N., Keller, M., et al. “How virtualization, decentralization and network building change the manufacturing landscape: An Industry 4.0 Perspective”. In: *International Journal of Science, Engineering and Technology* 8 (2014), pp. 37–44.

[5] Bourne, V. *After The Fall: Cost, Causes and Consequences of Unplanned Downtime.* 2018. URL: https://lp.servicemax.com/Vanson-Bourne-Whitepaper-Unplanned-Downtime-LP.html (visited on 2018-11-05).

[6] Burke, R., Mussomeli, A., Laaper, S., et al. *The smart factory: Responsive, adaptive, connected manufacturing.* Ed. by Deloitte Insights. 2017. URL: https://www2.deloitte.com/insights/us/en/focus/industry-4-0/smart-factory-connected-manufacturing.html (visited on 2018-11-05).

[7] Milan, A., Leal-Taixe, L., Reid, I., et al. *MOT16: A Benchmark for Multi-Object Tracking.* 2016. URL: http://arxiv.org/pdf/1603.00831v2.

[8] Redmon, J. and Farhadi, A. *YOLOv3: An Incremental Improvement.* 2018. URL: http://arxiv.org/pdf/1804.02767v1.

[9] Ren, S., He, K., Girshick, R., et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* 2015. URL: http://arxiv.org/pdf/1506.01497v3.

[10] Lin, T.-Y., Goyal, P., Girshick, R., et al. *Focal Loss for Dense Object Detection.* 2017. URL: http://arxiv.org/pdf/1708.02002v2.

[11] Liu, W., Anguelov, D., Erhan, D., et al. “SSD: Single Shot MultiBox Detector”. In: *(Keine Angabe)* 9905 (2016), pp. 21–37. ISSN: 0302-9743. DOI: 10.1007/978-3-319-46448-0_2. URL: http://arxiv.org/pdf/1512.02325v5.

[12] Bolme, D., Beveridge, J. R., Draper, B. A., et al. “Visual object tracking using adaptive correlation filters”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010.* Piscataway, NJ: IEEE, 2010, pp. 2544–2550. ISBN: 978-1-4244-6984-0. DOI: 10.1109/CVPR.2010.5539960.

[13] Lukežič, A., Vojíř, T., Čehovin, L., et al. “Discriminative Correlation Filter with Channel and Spatial Reliability”. In: *International Journal of Computer Vision* 126.7 (2018), pp. 671–688. ISSN: 0920-5691. DOI: 10.1007/s11263-017-1061-3. URL: http://arxiv.org/pdf/1611.08461v2.

[14] Wang, Q., Gao, J., Xing, J., et al. *DCFNet: Discriminant Correlation Filters Network for Visual Tracking.* 2017. URL: http://arxiv.org/pdf/1704.04057v1.

[15] Danelljan, M., Bhat, G., Khan, F. S., et al. *ECO: Efficient Convolution Operators for Tracking.* 2017. URL: http://arxiv.org/pdf/1611.09224v2.

[16] Brian, M. *Los Angeles Dog Photography.* 2018. URL: https://www.michaelbrianphoto.com/PEOPLE/Enjoy-Life/42 (visited on 2018-12-12).

[17] Qednau, R. *Narrow Roads Are Better Than Crosswalks.* 2015. URL: https://www.strongtowns.org/journal/2015/8/20/narrow-roads-are-better-than-crosswalks (visited on 2019-12-12).

[18] Dalal, N. and Triggs, B. “Histograms of Oriented Gradients for Human Detection”. In: *CVPR 2005.* Ed. by Schmid, C., Tomasi, C., and Soatto, S. Los Alamitos, Calif: IEEE Computer Society, 2005, pp. 886–893. ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.177.

[19] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., et al. “Object detection with discriminatively trained part-based models”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.9 (2010), pp. 1627–1645. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2009.167.

[20] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., et al. “Selective Search for Object Recognition”. In: *International Journal of Computer Vision* 104.2 (2013), pp. 154–171. ISSN: 0920-5691. DOI: 10.1007/s11263-013-0620-5.

[21] Lowe, D. G. “Object recognition from local scale-invariant features”. In: *The proceedings of the seventh IEEE international conference on computer vision.* s.l.: IEEE Computer Society, 1999, 1150–1157 vol.2. ISBN: 0-7695-0164-8. DOI: 10.1109/ICCV.1999.790410.

[22] Papageorgiou, C. P., Oren, M., and Poggio, T. “A general framework for object detection”. In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271).* Narosa Publishing House, 1998,

pp. 555–562. ISBN: 81-7319-221-9. DOI: 10.1109/ICCV.1998.710772.

[23] Donahue, J., Jia, Y., Vinyals, O., et al. *DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition.* 2013. URL: http://arxiv.org/pdf/1310.1531v1.

[24] Hosang, J., Benenson, R., and Schiele, B. *Learning non-maximum suppression.* 2017. URL: http://arxiv.org/pdf/1705.02950v2.

[25] Bodla, N., Singh, B., Chellappa, R., et al. *Soft-NMS – Improving Object Detection With One Line of Code.* 2017. URL: http://arxiv.org/pdf/1704.04503v2.

[26] Prokudin, S., Kappler, D., Nowozin, S., et al. "Learning to Filter Object Detections". In: *Pattern Recognition.* Ed. by Roth, V. and Vetter, T. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017. ISBN: 978-3-319-66709-6.

[27] Hosang, J., Benenson, R., and Schiele, B. *A convnet for non-maximum suppression.* 2015. URL: http://arxiv.org/pdf/1511.06437v3.

[28] Rosebrock, A. *A gentle guide to deep learning object detection.* 2018. URL: https://www.pyimagesearch.com/2018/05/14/a-gentle-guide-to-deep-learning-object-detection/ (visited on 2018-08-17).

[29] Tahir, A., Azam, S., Sagabala, S., et al. "Single object tracking system using fast compressive tracking". In: *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia).* IEEE, 2016, pp. 1–3. ISBN: 978-1-5090-2743-9. DOI: 10.1109/ICCE-Asia.2016.7804760.

[30] Lee, J., Iwana, B. K., Ide, S., et al. *Globally Optimal Object Tracking with Fully Convolutional Networks.* 2016. URL: http://arxiv.org/pdf/1612.08274v1.

[31] Luo, W., Xing, J., Milan, A., et al. *Multiple Object Tracking: A Literature Review.* 2017. URL: http://arxiv.org/pdf/1409.7618v4.

[32] Berclaz, J., Fleuret, F., Türetken, E., et al. "Multiple Object Tracking Using K-Shortest Paths Optimization". In: *IEEE transactions on pattern analysis and machine intelligence* 33.9 (2011), pp. 1806–1819. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2011.21.

[33] Leal-Taixé, L., Milan, A., Schindler, K., et al. *Tracking the Trackers: An Analysis of the State of the Art in Multiple Object Tracking.* 2017. URL: http://arxiv.org/pdf/1704.02781v1.

[34] Perera, A., Srinivas, C., Hoogs, A., et al. "Multi-Object Tracking Through Simultaneous Long Occlusions and Split-Merge Conditions". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06).* IEEE, 2006, pp. 666–673. ISBN: 0-7695-2597-0. DOI: 10.1109/CVPR.2006.195.

[35] Yang, C., Duraiswami, R., and Davis, L. "Fast multiple object tracking via a hierarchical particle filter". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1.* IEEE, 2005, 212–219 Vol. 1. ISBN: 0-7695-2334-X. DOI: 10.1109/ICCV.2005.95.

[36] Vo, B.-N., Mallick, M., Bar-Shalom, Y., et al. *Multitarget Tracking.* Ed. by Wiley Encyclopedia of Electrical and Electronics Engineering. 2015.

[37] Wang, N., Shi, J., Yeung, D.-Y., et al. *Understanding and Diagnosing Visual Tracking Systems.* 2015. URL: http://arxiv.org/pdf/1504.06055v1.

[38] Krebs, S., Duraisamy, B., and Flohr, F. "A survey on leveraging deep neural networks for object tracking". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC).* IEEE, 2017, pp. 411–418. ISBN: 978-1-5386-1526-3. DOI: 10.1109/ITSC.2017.8317904.

[39] Fiaz, M., Mahmood, A., and Jung, S. K. *Tracking Noisy Targets: A Review of Recent Object Tracking Approaches.* 2018. URL: http://arxiv.org/pdf/1802.03098v2.

[40] Cannons, K. *A Review of Visual Tracking: Tech. Rep. CSE-2008-07.* York Univ., Toronto, Canada, 2008.

[41] Smeulders, A. W. M., Chu, D. M., Cucchiara, R., et al. "Visual Tracking: An Experimental Survey". In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2014), pp. 1442–1468. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2013.230.

[42] Chen, Z., Hong, Z., and Tao, D. *An Experimental Survey on Correlation Filter-based Tracking.* 2015. URL: http://arxiv.org/pdf/1509.05520v1.

[43] Bar-Shalom, Y., Daum, F., and Huang, J. "The probabilistic data association filter". In: *IEEE Control Systems* 29.6 (2009), pp. 82–100. DOI: 10.1109/MCS.2009.934469.

[44] Blackman, S. S. "Multiple hypothesis tracking for multiple target tracking". In: *IEEE Aerospace and Electronic Systems Magazine* 19.1 (2004), pp. 5–18. ISSN: 0885-8985. DOI: 10.1109/MAES.2004.1263228.

[45] Maggio, E., Taj, M., and Cavallaro, A. "Efficient Multitarget Visual Tracking Using Random Finite Sets". In: *IEEE Transactions on Circuits and Systems for Video Technology* 18.8 (2008), pp. 1016–1027. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2008.928221.

[46] Tao, R., Gavves, E., and Smeulders, A. W. M. *Siamese Instance Search for Tracking.* 2016. URL: http://arxiv.org/pdf/1605.05863v1.

[47] Leal-Taixé, L., Ferrer, C. C., and Schindler, K. *Learning by tracking: Siamese CNN for robust target association.* 2014. URL: http://arxiv.org/pdf/1604.07866v3.

[48] Varior, R. R., Shuai, B., Lu, J., et al. *A Siamese Long Short-Term Memory Architecture for Human Re-Identification.* 2016. URL: http://arxiv.org/pdf/1607.08381v1.

[49] Bradski, G. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000). URL: https://docs.opencv.org (visited on 2018-08-01).

[50] Breitenstein, M. D., Reichlin, F., Leibe, B., et al. "Robust tracking-by-detection using a detector confidence particle filter". In: *2009 IEEE 12th International Conference on Computer Vision.* IEEE, 2009, pp. 1515–1522. ISBN: 978-1-4244-4420-5. DOI: 10.1109/ICCV.2009.5459278.

[51] Murray, S. *Real-Time Multiple Object Tracking - A Study on the Importance of Speed.* 2017. url: http://arxiv.org/pdf/1709.03572v2.

[52] Chahyati, D., Fanany, M. I., and Arymurthy, A. M. "Tracking People by Detection Using CNN Features". In: *Procedia Computer Science* 124 (2017), pp. 167–172. issn: 18770509. doi: 10.1016/j.procs.2017.12.143.

[53] Wu, B. and Nevatia, R. "Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors". In: *International Journal of Computer Vision* 75.2 (2007), pp. 247–266. issn: 0920-5691. doi: 10.1007/s11263-006-0027-7.

[54] Kuhn, H. W. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97. issn: 00281441. doi: 10.1002/nav.3800020109.

[55] Berclaz, J., Fleuret, F., and Fua, P. "Multiple object tracking using flow linear programming". In: *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance.* Piscataway: IEEE, 2009, pp. 1–8. isbn: 978-1-4244-5503-4. doi: 10.1109/PETS-WINTER.2009.5399488.

[56] Jiang, H., Fels, S., and Little, J. J. "A Linear Programming Approach for Multiple Object Tracking". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2007, pp. 1–8. isbn: 1-4244-1179-3. doi: 10.1109/CVPR.2007.383180.

[57] Chopra, S., Hadsell, R., and LeCun, Y. "Learning a Similarity Metric Discriminatively, with Application to Face Verification". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05).* IEEE, 2005, pp. 539–546. isbn: 0-7695-2372-2. doi: 10.1109/CVPR.2005.202.

[58] Porikli, F. and Yilmaz, A. "Object Detection and Tracking". In: *Video Analytics for Business Intelligence.* Ed. by Shan, C., Porikli, F., Xiang, T., et al. Vol. 409. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–41. isbn: 978-3-642-28597-4. doi: 10.1007/978-3-642-28598-1_1.

[59] Luo, H., Xie, W., Wang, X., et al. *Detect or Track: Towards Cost-Effective Video Object Detection/Tracking.* 2018. url: http://arxiv.org/pdf/1811.05340v1.

[60] Natarajan, P., Atrey, P. K., and Kankanhalli, M. "Multi-Camera Coordination and Control in Surveillance Systems". In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 11.4 (2015), pp. 1–30. issn: 15516857. doi: 10.1145/2710128.

[61] Räty, T. D. "Survey on Contemporary Remote Surveillance Systems for Public Safety". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40.5 (2010), pp. 493–515. issn: 1094-6977. doi: 10.1109/TSMCC.2010.2042446.

[62] Nguyen, C., Feng, W.-c., and Liu, F. "Hotspot: Making computer vision more effective for human video surveillance". In: *Information Visualization* 15.4 (2016), pp. 273–285. issn: 1473-8716. doi: 10.1177/1473871616630015.

[63] Tian, Y.-l., Brown, L., Hampapur, A., et al. "IBM smart surveillance system (S3): Event based video surveillance system with an open and extensible framework". In: *Machine Vision and Applications* 19.5-6 (2008), pp. 315–327. issn: 0932-8092. doi: 10.1007/s00138-008-0153-z.

[64] Leo, C. d. and Manjunath, B. S. "Multicamera video summarization and anomaly detection from activity motifs". In: *ACM Transactions on Sensor Networks* 10.2 (2014), pp. 1–30. issn: 15504859. doi: 10.1145/2530285.

[65] Girgensohn, A., Shipman, F., Dunnigan, A., et al. "Support for effective use of multiple video streams in security". In: *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks - VSSN '06.* Ed. by Aggarwal, J. K., Cucchiara, R., and Prati, A. New York, New York, USA: ACM Press, 2006, p. 19. isbn: 1595934960. doi: 10.1145/1178782.1178787.

[66] Ivanov, Y. A. and Wren, C. R. "Toward spatial queries for spatial surveillance tasks". In: *In Pervasive: Workshop on Pervasive Technology Applied to Real-World Experiences with RFID and Sensor Networks.* 2006.

[67] Wang, Y.-K., Fan, C.-T., and Huang, C.-R. "A Large Scale Video Surveillance System with Heterogeneous Information Fusion and Visualization for Wide Area Monitoring". In: *2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing.* IEEE, 2012, pp. 178–181. isbn: 978-1-4673-1741-2. doi: 10.1109/IIH-MSP.2012.49.

[68] Pham, T. V., Worring, M., and Smeulders, A. W. "A Multi-Camera Visual Surveillance System for Tracking of Reoccurrences of People". In: *2007 First ACM/IEEE International Conference on Distributed Smart Cameras.* IEEE, 2007, pp. 164–169. isbn: 978-1-4244-1353-9. doi: 10.1109/ICDSC.2007.4357520.

[69] Thalmann, D., Salamin, P., Ott, R., et al. "Advanced Mixed Reality Technologies for Surveillance and Risk Prevention Applications". In: *Computer and Information Sciences – ISCIS 2006.* Ed. by Hutchison, D., Kanade, T., Kittler, J., et al. Vol. 4263. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 13–23. isbn: 978-3-540-47242-1. doi: 10.1007/11902140_2.

[70] Ott, R., Gutiérrez, M., Thalmann, D., et al. "Advanced virtual reality technologies for surveillance and security applications". In: *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications - VRCIA '06.* Ed. by Sun, H. New York, New York, USA: ACM Press, 2006, p. 163. isbn: 1595933247. doi: 10.1145/1128923.1128949.

[71] Girgensohn, A., Kimber, D., Vaughan, J., et al. "DOTS: support for effective video surveillance". In: *Proceedings of the 15th ACM international conference on Multimedia.* ACM, 2007, pp. 423–432.

[72] KANADE, T., COLLINS, R., LIPTON, A., et al., eds. *Advances in Cooperative Multi-Sensor Video Surveillance.* Morgan Kaufmann, 1998.

[73] DEMEULEMEESTER, A., HOLLEMEERSCH, C.-F., LAMBERT, P., et al. "Demo: Real-time 3D visualization of multi-camera room occupancy monitoring for immersive communication systems". In: *2011 Fifth ACM/IEEE International Conference on Distributed Smart Cameras.* IEEE, 2011, pp. 1–2. ISBN: 978-1-4577-1708-6. DOI: `10.1109/ICDSC.2011.6042956`.

[74] FLECK, S., BUSCH, F., BIBER, P., et al. "3D Surveillance A Distributed Network of Smart Cameras for Real-Time Tracking and its Visualization in 3D". In: *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06).* IEEE, 2006, p. 118. ISBN: 0-7695-2646-2. DOI: `10.1109/CVPRW.2006.6`.

[75] CORRAL-SOTO, E. R., TAL, R., WANG, L., et al. "3D Town: The Automatic Urban Awareness Project". In: *2012 Ninth Conference on Computer and Robot Vision.* IEEE, 2012, pp. 433–440. ISBN: 978-1-4673-1271-4. DOI: `10.1109/CRV.2012.64`.

[76] XING, Y., NAGAHASHI, H., and ZHANG, X. "A 3D Dynamic Visualization Surveillance System". In: *International Journal of Computer Science Issues* 13.5 (2016), pp. 36–44. ISSN: 16940814. DOI: `10.20943/01201605.3644`.

[77] SANKARANARAYANAN, A. C., PATRO, R., TURAGA, P., et al. "Modeling and Visualization of Human Activities for Multicamera Networks". In: *EURASIP Journal on Image and Video Processing* 2009.4 (2009), pp. 1–13. ISSN: 1687-5176. DOI: `10.1155/2009/259860`.

[78] BRANDLE, N., MATYUS, T., BRUNNHUBER, M., et al. "Realistic Interactive Pedestrian Simulation and Visualization for Virtual 3D Environments". In: *2009 15th International Conference on Virtual Systems and Multimedia.* IEEE, 2009, pp. 179–184. ISBN: 978-0-7695-3790-0. DOI: `10.1109/VSMM.2009.33`.

[79] YANG, Y., CHANG, M.-C., TU, P., et al. "Seeing as it happens: Real time 3D video event visualization". In: *2015 IEEE International Conference on Image Processing (ICIP).* IEEE, 2015, pp. 2875–2879. ISBN: 978-1-4799-8339-1. DOI: `10.1109/ICIP.2015.7351328`.

[80] XIE, Y., WANG, M., LIU, X., et al. "Surveillance Video Synopsis in GIS". In: *ISPRS International Journal of Geo-Information* 6.11 (2017), p. 333. ISSN: 2220-9964. DOI: `10.3390/ijgi6110333`.

[81] KUMAR, A., CHAVAN, P. S., SHARATCHANDRA, V. K., et al. "3D Estimation and Visualization of Motion in a Multicamera Network for Sports". In: *2011 Irish Machine Vision and Image Processing Conference.* IEEE, 2011, pp. 15–19. ISBN: 978-0-7695-4629-2. DOI: `10.1109/IMVIP.2011.12`.

[82] ROTH, P. M., SETTGAST, V., WIDHALM, P., et al. "Next-generation 3D visualization for visual surveillance". In: *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS).* IEEE, 2011, pp. 343–348. ISBN: 978-1-4577-0844-2. DOI: `10.1109/AVSS.2011.6027348`.

[83] DECAMP, P., SHAW, G., KUBAT, R., et al. "An immersive system for browsing and visualizing surveillance video". In: *Proceedings of the international conference on Multimedia - MM '10.* Ed. by DEL BIMBO, A., CHANG, S.-F., and SMEULDERS, A. New York, New York, USA: ACM Press, 2010, p. 371. ISBN: 9781605589336. DOI: `10.1145/1873951.1874002`.

[84] HAAN, G. de, SCHEUER, J., VRIES, R. de, et al. "Egocentric navigation for video surveillance in 3D Virtual Environments". In: *2009 IEEE Symposium on 3D User Interfaces.* IEEE, 2009, pp. 103–110. ISBN: 978-1-4244-3965-2. DOI: `10.1109/3DUI.2009.4811214`.

[85] KIM, K., OH, S., LEE, J., et al. "Augmenting Aerial Earth Maps with dynamic information". In: *2009 8th IEEE International Symposium on Mixed and Augmented Reality.* IEEE, 2009, pp. 35–38. ISBN: 978-1-4244-5390-0. DOI: `10.1109/ISMAR.2009.5336505`.

[86] WANG, Y., BOWMAN, D., KRUM, D., et al. "Effects of video placement and spatial context presentation on path reconstruction tasks with contextualized videos". In: *IEEE transactions on visualization and computer graphics* 14.6 (2008), pp. 1755–1762. ISSN: 1077-2626. DOI: `10.1109/TVCG.2008.126`.

[87] WANG, Y., KRUM, D. M., COELHO, E. M., et al. "Contextualized videos: Combining videos with environment models to support situational understanding". In: *IEEE transactions on visualization and computer graphics* 13.6 (2007), pp. 1568–1575. ISSN: 1077-2626. DOI: `10.1109/TVCG.2007.70544`.

[88] SEBE, I. O., HU, J., YOU, S., et al. "3D video surveillance with Augmented Virtual Environments". In: *First ACM SIGMM international workshop on Video surveillance - IWVS '03.* Ed. by CHANG, E. Y. and WANG, Y.-F. New York, New York, USA: ACM Press, 2003, p. 107. ISBN: 158113780X. DOI: `10.1145/982452.982466`.

[89] HALL, B. and TRIVEDI, M. M. "A Novel Graphical Interface and Context Aware Map for Incident Detection and Monitoring". In: *9th World Congress on Intelligent Transport Systems.* Chicago, Illinois, 2002.

[90] SAWHNEY, H. S., ARPA, A., KUMAR, R., et al. *Video Flashlights - Real Time Rendering of Multiple Videos for Immersive Model Visualization.* 2002. DOI: `10.2312/EGWR/EGWR02/157-168`.

[91] CHANDRAJIT, M., GIRISHA, R., and VASUDEV, T. "Multiple Objects Tracking in Surveillance Video Using Color and Hu Moments". In: *Signal & Image Processing : An International Journal* 7.3 (2016), pp. 15–27. ISSN: 22293922. DOI: `10.5121/sipij.2016.7302`.

[92] GRUENWEDEL, S., JELACA, V., NINO-CASTANEDA, J. O., et al. "Low-complexity scalable distributed multicamera tracking of humans". In: *ACM Transactions on Sensor Networks* 10.2 (2014), pp. 1–32. ISSN: 15504859. DOI: `10.1145/2530282`.

[93] JIANG, X., RODNER, E., and DENZLER, J. "Multi-person Tracking-by-Detection Based on Calibrated Multi-camera Systems". In: *Computer Vision and Graphics.* Ed. by HUTCHISON, D., KANADE, T., KITTLER, J., et al. Vol. 7594. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 743–751. ISBN: 978-3-642-33563-1. DOI: `10.1007/978-3-642-33564-8_89`.

[94] Yildiz, A. and Akgul, Y. S. “A Fast Method for Tracking People with Multiple Cameras”. In: *Trends and Topics in Computer Vision.* Ed. by Hutchison, D., Kanade, T., Kittler, J., et al. Vol. 6553. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 128–138. isbn: 978-3-642-35748-0. doi: 10.1007/978-3-642-35749-7_10.

[95] Sternig, S., Mauthner, T., Irschara, A., et al. “Multi-camera multi-object tracking by robust hough-based homography projections”. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops).* IEEE, 2011, pp. 1689–1696. isbn: 978-1-4673-0063-6. doi: 10.1109/ICCVW.2011.6130453.

[96] Eshel, R. and Moses, Y. “Tracking in a Dense Crowd Using Multiple Cameras”. In: *International Journal of Computer Vision* 88.1 (2010), pp. 129–143. issn: 0920-5691. doi: 10.1007/s11263-009-0307-0.

[97] Khan, S. M. and Shah, M. “Tracking multiple occluding people by localizing on multiple scene planes”. In: *IEEE transactions on pattern analysis and machine intelligence* 31.3 (2009), pp. 505–519. issn: 0162-8828. doi: 10.1109/TPAMI.2008.102.

[98] Muñoz-Salinas, R., Medina-Carnicer, R., Madrid-Cuevas, F. J., et al. “Multi-camera people tracking using evidential filters”. In: *International Journal of Approximate Reasoning* 50.5 (2009), pp. 732–749. doi: 10.1016/j.ijar.2009.02.001.

[99] Kanade, T., Collins, R., Lipton, A., et al., eds. *Cooperative Multi-Sensor Video Surveillance.* 1997.

[100] Collins, R. T., Lipton, A. J., and Kanade, T. “A System for Video Surveillance and Monitoring”. In: 1999.

[101] Collins, R. T., Lipton, A. J., Fujiyoshi, H., et al. “Algorithms for cooperative multisensor surveillance”. In: *Proceedings of the IEEE* 89.10 (2001), pp. 1456–1477. issn: 00189219. doi: 10.1109/5.959341.

[102] Bobick, A. F. and Davis, J. W. “The recognition of human movement using temporal templates”. In: *IEEE transactions on pattern analysis and machine intelligence* 23.3 (2001), pp. 257–267. issn: 0162-8828. doi: 10.1109/34.910878.

[103] Rieffel, E. G., Girgensohn, A., Kimber, D., et al. “Geometric Tools for Multicamera Surveillance Systems”. In: *2007 First ACM/IEEE International Conference on Distributed Smart Cameras.* IEEE, 2007, pp. 132–139. isbn: 978-1-4244-1353-9. doi: 10.1109/ICDSC.2007.4357516.

[104] Yang, T., Chen, F., Kimber, D., et al. “Robust People Detection and Tracking in a Multi-Camera Indoor Visual Surveillance System”. In: *Multimedia and Expo, 2007 IEEE International Conference on.* IEEE, 2007, pp. 675–678. isbn: 1-4244-1016-9. doi: 10.1109/ICME.2007.4284740.

[105] Girgensohn, A., Shipman, F., Turner, T., et al. “Effects of presenting geographic context on tracking activity between cameras”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI ’07.* Ed. by Rosson, M. B. and Gilmore, D. New York, New York, USA: ACM Press, 2007, p. 1167. isbn: 9781595935939. doi: 10.1145/1240624.1240801.

[106] Kimber, D., Dunnigan, T., Girgensohn, A., et al. “Trailblazing: Video Playback Control by Direct Object Manipulation”. In: *Multimedia and Expo, 2007 IEEE International Conference on.* IEEE, 2007, pp. 1015–1018. isbn: 1-4244-1016-9. doi: 10.1109/ICME.2007.4284825.

[107] Xie, Y., Wang, M., Liu, X., et al. “Integration of GIS and Moving Objects in Surveillance Video”. In: *ISPRS International Journal of Geo-Information* 6.4 (2017), p. 94. issn: 2220-9964. doi: 10.3390/ijgi6040094.

[108] Jancosek, M. and Pajdla, T. “Multi-view reconstruction preserving weakly-supported surfaces”. In: *CVPR 2011.* IEEE, 2011, pp. 3121–3128. isbn: 978-1-4577-0394-2. doi: 10.1109/CVPR.2011.5995693.

[109] Irschara, A., Zach, C., and Bischof, H. “Towards Wiki-based Dense City Modeling”. In: *2007 IEEE 11th International Conference on Computer Vision.* IEEE, 2007, pp. 1–8. isbn: 978-1-4244-1630-1. doi: 10.1109/ICCV.2007.4409216.

[110] Girgensohn, A., Dunnigan, A. E., Shipman, F. M., et al. “Interface for browsing and viewing video from multiple cameras simultaneously that conveys spatial and temporal proximity”. US 8,274,564. 2012.

[111] Zhou, H., Liu, Q., Kimber, D., et al. “System and Method for User Monitoring Interface of 3D Video Streams from Multiple Cameras”. US 7,944,454. 2011.

[112] Oracle. *MySQL Reference Manual 8.0.* 2018. url: https://dev.mysql.com/doc/refman/8.0/en/ (visited on 2018-10-08).

[113] Ippolito, B. and Preston, D. *Eventlet 0.24.1 Documentation.* 2010. url: http://eventlet.net/doc/ (visited on 2018-10-08).

[114] Ronacher, A. *Flask 1.0 User Guide.* 2018. url: http://flask.pocoo.org/docs/1.0/ (visited on 2018-10-08).

[115] Grinberg, M. *Flask-SocketIO 3.0.2 Documentation.* 2018. url: https://flask-socketio.readthedocs.io/en/latest/ (visited on 2018-10-08).

[116] Sanfilippo, S. *Redis 4.0 Documentation.* 2018. url: https://redis.io/documentation (visited on 2018-10-08).

[117] Bewley, A., Ge, Z., Ott, L., et al. “Simple Online and Realtime Tracking”. In: (2016), pp. 3464–3468. doi: 10.1109/ICIP.2016.7533003. url: http://arxiv.org/pdf/1602.00763v2.

[118] Wojke, N., Bewley, A., and Paulus, D. *Simple Online and Realtime Tracking with a Deep Association Metric.* 2017. url: http://arxiv.org/pdf/1703.07402v1.

[119] Huang, J., Rathod, V., Sun, C., et al. *Speed/accuracy trade-offs for modern convolutional object detectors.* 2017. url: http://arxiv.org/pdf/1611.10012v3.

[120] Mahmoud, Z. *Experimenting with SORT: GitHub repository.* 2017. url: https://github.com/ZidanMusk/experimenting-with-sort/blob/master/data_association.py (visited on 2018-10-12).

[121] JONES, E., OLIPHANT, T., and PETERSON, P. *SciPy: Open Source Scientific Tools for Python*. 2001. URL: http://www.scipy.org/ (visited on 2018-10-15).

[122] BURKARD, R. E. and ÇELA, E. "Linear Assignment Problems and Extensions". In: *Handbook of Combinatorial Optimization*. Ed. by DU, D.-Z. and PARDALOS, P. M. Boston, MA: Springer US, 1999, pp. 75–149. ISBN: 978-1-4419-4813-7. DOI: 10.1007/978-1-4757-3023-4_2.

[123] CANTOR, G. "Ein Beitrag zur Mannigfaltigkeitslehre". In: *Journal für die reine und angewandte Mathematik (Crelles Journal)* 1878.84 (1878), pp. 242–258. ISSN: 0075-4102. DOI: 10.1515/crll.1878.84.242.

[124] ORACLE. *MySQL Connector: Python Developer Guide*. 2018. URL: https://dev.mysql.com/doc/connector-python/en/ (visited on 2018-10-16).

[125] CABELLO, R. *three.js Javascript 3D Library Documentation*. 2018. URL: https://threejs.org/docs/index.html (visited on 2018-10-18).

[126] KHRONOS GROUP. *WebGL Overview*. 2018. URL: https://www.khronos.org/webgl/ (visited on 2018-10-18).

[127] BOSTOCK, M. *D3.js: Data-Driven Documents*. 2017. URL: https://d3js.org/.

[128] SCHULZRINNE, H., RAO, A., LANPHIER, R., et al. *Real-Time Streaming Protocol Version 2.0*. 2016. DOI: 10.17487/RFC7826. URL: https://rfc-editor.org/rfc/rfc7826.txt.

[129] KANNALA, J. and BRANDT, S. S. "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses". In: *IEEE transactions on pattern analysis and machine intelligence* 28.8 (2006), pp. 1335–1340. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2006.153.

[130] *OpenCV Fisheye Camera Model*. 2018. URL: https://docs.opencv.org/trunk/db/d58/group__calib3d__fisheye.html (visited on 2018-07-30).

[131] SANKARANARAYANAN, A. C. and CHELLAPPA, R. "Optimal Multi-View Fusion of Object Locations". In: *2008 IEEE Workshop on Motion and video Computing*. IEEE, 2008, pp. 1–8. ISBN: 978-1-4244-2000-1. DOI: 10.1109/WMVC.2008.4544048.

[132] ESHEL, R. and MOSES, Y. "Homography based multiple camera detection and tracking of people in a dense crowd". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8. ISBN: 978-1-4244-2242-5. DOI: 10.1109/CVPR.2008.4587539.

[133] KHAN, S. M. and SHAH, M. "A Multiview Approach to Tracking People in Crowded Scenes Using a Planar Homography Constraint". In: *Computer Vision – ECCV 2006*. Ed. by LEONARDIS, A., BISCHOF, H., and PINZ, A. Vol. 3954. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 133–146. ISBN: 978-3-540-33838-3. DOI: 10.1007/11744085_11.

[134] KIM, K. and DAVIS, L. S. "Multi-camera Tracking and Segmentation of Occluded People on Ground Plane Using Search-Guided Particle Filtering". In: *Computer Vision – ECCV 2006*. Ed. by HUTCHISON, D., KANADE, T., KITTLER, J., et al. Vol. 3953. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 98–109. ISBN: 978-3-540-33836-9. DOI: 10.1007/11744078_8.

[135] LEVENBERG, K. "A method for the solution of certain non-linear problems in least squares". In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168. DOI: 10.1090/qam/10666.

[136] MARQUARDT, D. W. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441. ISSN: 0368-4245. DOI: 10.1137/0111030.

[137] ROUSSEEUW, P. J. "Least Median of Squares Regression". In: *Journal of the American Statistical Association* 79.388 (1984), pp. 871–880. ISSN: 0162-1459. DOI: 10.1080/01621459.1984.10477105.

[138] NVIDIA CORPORATION. *TensorRT 5.0 Developer Guide*. 2018. URL: https://docs.nvidia.com/deeplearning/sdk/tensorrt-developer-guide/index.html (visited on 2018-10-20).

[139] TAN, L., WANG, Y., YU, H., et al. "Automatic Camera Calibration Using Active Displays of a Virtual Pattern". In: *Sensors (Basel, Switzerland)* 17.4 (2017). DOI: 10.3390/s17040685.

[140] FORSTER, F. "Camera calibration: Active versus passive targets". In: *Optical Engineering* 50.11 (2011), p. 113601. ISSN: 0091-3286. DOI: 10.1117/1.3643726.

[141] HERRERA, D., KANNALA, C. J., and HEIKKILA, J. "Forget the checkerboard: Practical self-calibration using a planar scene". In: *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Piscataway, NJ: IEEE, 2016, pp. 1–9. ISBN: 978-1-5090-0641-0. DOI: 10.1109/WACV.2016.7477641.

[142] GIRSHICK, R., DONAHUE, J., DARRELL, T., et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2013. URL: http://arxiv.org/pdf/1311.2524v5.

[143] ZHAI, M., ROSHTKHARI, M. J., and MORI, G. *Deep Learning of Appearance Models for Online Object Tracking*. 2016. URL: http://arxiv.org/pdf/1607.02568v1.

[144] SHARMA, S., ANSARI, J. A., MURTHY, J. K., et al. *Beyond Pixels: Leveraging Geometry and Shape Cues for Online Multi-Object Tracking*. 2018. URL: http://arxiv.org/pdf/1802.09298v2.

[145] GORDON, D., FARHADI, A., and FOX, D. *Re3: Real-Time Recurrent Regression Networks for Visual Tracking of Generic Objects*. 2018. URL: http://arxiv.org/pdf/1705.06368v3.

[146] KRISTAN, M., MATAS, J., LEONARDIS, A., et al. "A Novel Performance Evaluation Methodology for Single-Target Trackers". In: *IEEE transactions on pattern analysis and machine intelligence* 38.11 (2016), pp. 2137–2155. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2516982. URL: http://arxiv.org/pdf/1503.01313v3.

[147] *Computer Vision Annotation Tool (CVAT): GitHub repository*. 2018. URL: https://github.com/opencv/cvat (visited on 2018-09-22).

[148] ČEHOVIN, L., LEONARDIS, A., and KRISTAN, M. "Visual object tracking performance measures revisited".

In: *IEEE Transactions on Image Processing* (2016), p. 1. ISSN: 1057-7149. DOI: `10.1109/TIP.2016.2520370`. URL: `http://arxiv.org/pdf/1502.05803v3`.

[149] Schuhmacher, D., Vo, B.-T., and Vo, B.-N. "A Consistent Metric for Performance Evaluation of Multi-Object Filters". In: *IEEE Transactions on Signal Processing* 56.8 (2008), pp. 3447–3457. DOI: `10.1109/TSP.2008.920469`.

[150] Edward, K. K., Matthew, P. D., and Michael, B. H. "An information theoretic approach for tracker performance evaluation". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1523–1529. ISBN: 978-1-4244-4420-5. DOI: `10.1109/ICCV.2009.5459275`.

[151] Smith, K., Gatica-Perez, D., Odobez, J., et al. "Evaluating Multi-Object Tracking". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*. IEEE, 2005, p. 36. ISBN: 0-7695-2372-2. DOI: `10.1109/CVPR.2005.453`.

[152] Bernardin, K. and Stiefelhagen, R. "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics". In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10. ISSN: 1687-5176. DOI: `10.1155/2008/246309`.

[153] Carvalho, P., Cardoso, J. S., and Corte-Real, L. "Filling the gap in quality assessment of video object tracking". In: *Image and Vision Computing* 30.9 (2012), pp. 630–640. ISSN: 02628856. DOI: `10.1016/j.imavis.2012.06.002`.

[154] Kasturi, R., Goldgof, D., Soundararajan, P., et al. "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol". In: *IEEE transactions on pattern analysis and machine intelligence* 31.2 (2009), pp. 319–336. ISSN: 0162-8828. DOI: `10.1109/TPAMI.2008.57`.

[155] Li, Y., Huang, C., and Nevatia, R. "Learning to associate: HybridBoosted multi-target tracker for crowded scene". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 2953–2960. ISBN: 978-1-4244-3992-8. DOI: `10.1109/CVPR.2009.5206735`.

[156] Leichter, I. and Krupka, E. "Monotonicity and error type differentiability in performance measures for target detection and tracking in video". In: *IEEE transactions on pattern analysis and machine intelligence* 35.10 (2013), pp. 2553–2560. ISSN: 0162-8828. DOI: `10.1109/TPAMI.2013.70`.

[157] Manohar, V., Soundararajan, P., Raju, H., et al. "Performance Evaluation of Object Detection and Tracking in Video". In: *Computer Vision – ACCV 2006*. Ed. by Hutchison, D., Kanade, T., Kittler, J., et al. Vol. 3852. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 151–161. ISBN: 978-3-540-31244-4. DOI: `10.1007/11612704_16`.

[158] Bashir, F. and Porikli, F. *Performance Evaluation of Object Detection and Tracking Systems*. 2006.

[159] Godil, A., Bostelman, R., Shackleford, W., et al. *Performance Metrics for Evaluating Object and Human Detection and Tracking Systems*. National Institute of Standards and Technology, 2014. DOI: `10.6028/NIST.IR.7972`.

[160] Wu, B. and Nevatia, R. "Tracking of Multiple, Partially Occluded Humans based on Static Body Part Detection". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*. IEEE, 2006, pp. 951–958. ISBN: 0-7695-2597-0. DOI: `10.1109/CVPR.2006.312`.

[161] Ristani, E., Solera, F., Zou, R. S., et al. *Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking*. 2016. URL: `http://arxiv.org/pdf/1609.01775v2`.

[162] Black, J., Ellis, T., and Rosin, P. "A Novel Method for Video Tracking Performance Evaluation". In: *In Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS*. 2003, pp. 125–132.

[163] Jaynes, C., Webb, S., Steele, R. M., et al. "An Open Development Environment for Evaluation of Video Surveillance Systems". In: *Proceedings of the Third International Workshop on Performance Evaluation of Tracking and Surveillance (PETS'2002*. 2002, pp. 32–39.

[164] Cavallaro, A. and F. Ziliani. "Characterisation of tracking performance". In: *Proceedings of Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*. 2005.

[165] Bernardin, K., Elbs, A., and Stiefelhagen, R. *Multiple Object Tracking Performance Metrics and Evaluation in a Smart Room Environment*. 2006.

[166] Brown, L. M., Senior, A. W., Tian, Y.-l., et al. "Performance evaluation of surveillance systems under varying conditions". In: *In: Proceedings of IEEE PETS Workshop*. 2005, pp. 1–8.

[167] Ellis, T. "Performance Metrics and Methods for Tracking in Surveillance". In: *Proceedings of the Third International Workshop on Performance Evaluation of Tracking and Surveillance*. 2002.

[168] Ibrahim, A. M., Shafie, A. A., and Rashid, M. M. "Performance Metrics in Video Surveillance System". In: *Journal of Engineering Science and Technology* 8.2 (2013), pp. 199–216.

[169] Geiger, A., Lenz, P., and Urtasun, R. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012*. Piscataway, NJ: IEEE, 2012, pp. 3354–3361. ISBN: 978-1-4673-1228-8. DOI: `10.1109/CVPR.2012.6248074`.

[170] Wen, L., Du Dawei, Cai, Z., et al. *UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking*. 2016. URL: `http://arxiv.org/pdf/1511.04136v3`.

[171] Stiefelhagen, R., Bernardin, K., Bowers, R., et al. "The CLEAR 2006 Evaluation". In: *Multimodal technologies for perception of humans*. Ed. by Stiefelhagen, R. Vol. 4122. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidel-

berg, 2007, pp. 1–44. ISBN: 978-3-540-69567-7. DOI: 10.1007/978-3-540-69568-4_1.

[172] Heindl, C. *Python MOT metrics package: GitHub repository.* 2018. URL: https://github.com/cheind/py-motmetrics (visited on 2018-09-21).

[173] Ryzhov, S. *Woman crossing the street at pedestrian crossing.* 2018. URL: https://stock.adobe.com/bg/search?serie_id=84780718&asset_id=84791571 (visited on 2018-12-12).

[174] Chen, W., Cao, L., Chen, X., et al. *An equalised global graphical model-based approach for multi-camera object tracking.* 2016. URL: http://arxiv.org/pdf/1502.03532v2.

[175] Zoph, B., Vasudevan, V., Shlens, J., et al. *Learning Transferable Architectures for Scalable Image Recognition.* 2017. URL: http://arxiv.org/pdf/1707.07012v4.

[176] Szegedy, C., Ioffe, S., Vanhoucke, V., et al. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.* 2016. URL: http://arxiv.org/pdf/1602.07261v2.

[177] He, K., Zhang, X., Ren, S., et al. *Deep Residual Learning for Image Recognition.* 2015. URL: http://arxiv.org/pdf/1512.03385v1.

[178] Dai, J., Li, Y., He, K., et al. *R-FCN: Object Detection via Region-based Fully Convolutional Networks.* 2016. URL: http://arxiv.org/pdf/1605.06409v2.

[179] Henriques, J. F., Caseiro, R., Martins, P., et al. "High-Speed Tracking with Kernelized Correlation Filters". In: *IEEE transactions on pattern analysis and machine intelligence* 37.3 (2015), pp. 583–596. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2014.2345390. URL: http://arxiv.org/pdf/1404.7584v3.

[180] Luo, H., Xie, W., Wang, X., et al. *Detect or Track: Towards Cost-Effective Video Object Detection/Tracking.* 2018. URL: http://arxiv.org/pdf/1811.05340v1.

[181] Ristani, E. and Tomasi, C. "Tracking Multiple People Online and in Real Time". In: *Computer Vision – ACCV 2014*. Ed. by Cremers, D., Reid, I., Saito, H., et al. Vol. 9007. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 444–459. ISBN: 978-3-319-16813-5. DOI: 10.1007/978-3-319-16814-2_29.

[182] Solera, F., Calderara, S., Ristani, E., et al. "Tracking Social Groups Within and Across Cameras". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.3 (2017), pp. 441–453. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2016.2607378.

[183] Bakker, C. de, Aries, M., Kort, H., et al. "Occupancy-based lighting control in open-plan office spaces: A state-of-the-art review". In: *Building and Environment* 112 (2017), pp. 308–321. ISSN: 03601323. DOI: 10.1016/j.buildenv.2016.11.042.

[184] Dong, J., Winstead, C., Nutaro, J., et al. "Occupancy-Based HVAC Control with Short-Term Occupancy Prediction Algorithms for Energy-Efficient Buildings". In: *Energies* 11.9 (2018), p. 2427. ISSN: 1996-1073. DOI: 10.3390/en11092427.

[185] Chen, Z., Jiang, C., and Xie, L. "Building occupancy estimation and detection: A review". In: *Energy and Buildings* 169 (2018), pp. 260–270. ISSN: 03787788. DOI: 10.1016/j.enbuild.2018.03.084.

[186] Labeodan, T., Zeiler, W., Boxem, G., et al. "Occupancy measurement in commercial office buildings for demand-driven control applications—A survey and detection system evaluation". In: *Energy and Buildings* 93 (2015), pp. 303–314. ISSN: 03787788. DOI: 10.1016/j.enbuild.2015.02.028.

[187] Kong, Y. and Fu, Y. *Human Action Recognition and Prediction: A Survey.* 2018. URL: http://arxiv.org/pdf/1806.11230v2.

[188] Herath, S., Harandi, M., and Porikli, F. *Going Deeper into Action Recognition: A Survey.* 2017. URL: http://arxiv.org/pdf/1605.04988v2.

[189] Subetha, T. and Chitrakala, S. "A survey on human activity recognition from videos". In: *2016 International Conference on Information Communication and Embedded Systems (ICICES)*. IEEE, 2016, pp. 1–7. ISBN: 978-1-5090-2552-7. DOI: 10.1109/ICICES.2016.7518920.

[190] Dhiman, C. and Vishwakarma, D. K. "A review of state-of-the-art techniques for abnormal human activity recognition". In: *Engineering Applications of Artificial Intelligence* 77 (2019), pp. 21–45. ISSN: 09521976. DOI: 10.1016/j.engappai.2018.08.014.

[191] Guthrie, J., Mishra, B. K., and Prasad, A. "Minimizing retail shrinkage due to employee theft". In: *International Journal of Retail & Distribution Management* 34.11 (2006), pp. 817–832. ISSN: 0959-0552. DOI: 10.1108/09590550610710228.

[192] Mufutau, G. O. "Fraud Detections, Preventions and Management in Stores/Warehouses in an Emerging Economy". In: *Innovative Systems Design and Engineering* 4.12 (2013), pp. 13–25.

[193] Cisco-Eagle. *Industrial & Warehouse Security: Put a Stop To Theft and Pilferage.* 2018. URL: http://www.cisco-eagle.com/uploads/White-Papers/eBook-Loss_Prevention-tall.pdf (visited on 2018-10-25).

[194] Wang, B., Li, H., Rezgui, Y., et al. "BIM based virtual environment for fire emergency evacuation". In: *TheScientificWorldJournal* 2014 (2014), p. 589016. DOI: 10.1155/2014/589016.

[195] Rueppel, U. and Stuebbe, K. M. "BIM-based indoor-emergency-navigation-system for complex buildings". In: *Tsinghua Science and Technology* 13.S1 (2008), pp. 362–367. ISSN: 1007-0214. DOI: 10.1016/S1007-0214(08)70175-5.

[196] Rausand, M. *Risk Assessment: Theory, Methods, and Applications.* Statistics in Practice. Hoboken: Wiley, 2013. ISBN: 978-0470637647. DOI: 10.1002/9781118281116. URL: http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=597754.

[197] Wang, M. and Deng, W. *Deep Face Recognition: A Survey.* 2018. URL: http://arxiv.org/pdf/1804.06655v7.

[198] Prihasto, B., Choirunnisa, S., Nurdiansyah, M. I., et al. "A survey of deep face recognition in the wild". In: *2016 International Conference on Orange Technologies (ICOT)*. IEEE, 2016, pp. 76–79. ISBN: 978-1-5386-4831-5. DOI: 10.1109/ICOT.2016.8278983.

[199] Jafri, R. and Arabnia, H. R. "A Survey of Face Recognition Techniques". In: *Journal of Information Processing Systems* 5.2 (2009), pp. 41–68. doi: 10.3745/JIPS.2009.5.2.041.

[200] Zou, X., Kittler, J., and Messer, K. "Illumination Invariant Face Recognition: A Survey". In: *2007 First IEEE International Conference on Biometrics: Theory, Applications, and Systems*. IEEE, 2007, pp. 1–8. isbn: 978-1-4244-1596-0. doi: 10.1109/BTAS.2007.4401921.

[201] Bowyer, K. W., Chang, K., and Flynn, P. "A survey of approaches and challenges in 3D and multimodal 3D+2D face recognition". In: *Computer Vision and Image Understanding* 101.1 (2006), pp. 1–15. issn: 10773142. doi: 10.1016/j.cviu.2005.05.005.

[202] Zhao, W., Chellappa, R., Phillips, P. J., et al. "Face recognition: A Literature Survey". In: *ACM Computing Surveys* 35.4 (2003), pp. 399–458. issn: 03600300. doi: 10.1145/954339.954342.

[203] Ganesharajah, T., Hall, N. G., and Sriskandarajah, C. "Design and operational issues in AGV-served manufacturing systems". In: *Annals of Operations Research* 76.0 (1998), pp. 109–154. issn: 02545330. doi: 10.1023/A:1018936219150.

[204] Andreasson, H., Bouguerra, A., Cirillo, M., et al. "Autonomous Transport Vehicles: Where We Are and What Is Missing". In: *IEEE Robotics & Automation Magazine* 22.1 (2015), pp. 64–75. issn: 1070-9932. doi: 10.1109/MRA.2014.2381357.

[205] Li, Q., Adriaansen, A. C., Udding, J. T., et al. "Design and Control of Automated Guided Vehicle Systems: A Case Study". In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 13852–13857. issn: 14746670. doi: 10.3182/20110828-6-IT-1002.01232.

[206] Syu, J.-L., Li, H.-T., Chiang, J.-S., et al. "A computer vision assisted system for autonomous forklift vehicles in real factory environment". In: *Multimedia Tools and Applications* 76.18 (2017), pp. 18387–18407. issn: 1380-7501. doi: 10.1007/s11042-016-4123-6.

[207] Hellstrom, T. and Ringdahl, O. "Follow the Past: A path-tracking algorithm for autonomous vehicles". In: *International Journal of Vehicle Autonomous Systems* 4.2/3/4 (2006), p. 216. issn: 1471-0226. doi: 10.1504/IJVAS.2006.012208.

[208] Wit, J., Crane, C. D., and Armstrong, D. "Autonomous ground vehicle path tracking". In: *Journal of Robotic Systems* 21.8 (2004), pp. 439–449. issn: 0741-2223. doi: 10.1002/rob.20031.

[209] Saputra, M. R. U., Markham, A., and Trigoni, N. "Visual SLAM and Structure from Motion in Dynamic Environments". In: *ACM Computing Surveys* 51.2 (2018), pp. 1–36. issn: 03600300. doi: 10.1145/3177853.

[210] Taketomi, T., Uchiyama, H., and Ikeda, S. "Visual SLAM algorithms: A survey from 2010 to 2016". In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (2017), p. 99. issn: 1882-6695. doi: 10.1186/s41074-017-0027-2.

[211] Bresson, G., Alsayed, Z., Yu, L., et al. "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving". In: *IEEE Transactions on Intelligent Vehicles* 2.3 (2017), pp. 194–220. issn: 2379-8904. doi: 10.1109/TIV.2017.2749181.

[212] Ros, G., Sappa, A. D., Ponsa, D., et al. "Visual SLAM for Driverless Cars: A Brief Survey". In: *Proceedings of the 2012 IEEE Intelligent Vehicles Symposium Workshops*. Piscataway, NJ: IEEE, 2012. isbn: 978-1-4673-2119-8.

[213] Fuentes-Pacheco, J., Ruiz-Ascencio, J., and Rendón-Mancha, J. M. "Visual simultaneous localization and mapping: A survey". In: *Artificial Intelligence Review* 43.1 (2015), pp. 55–81. issn: 0269-2821. doi: 10.1007/s10462-012-9365-8.

[214] Anavatti, S. G., Francis, S. L. X., and Garratt, M. "Path-planning modules for Autonomous Vehicles: Current status and challenges". In: *2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*. IEEE, 2015, pp. 205–214. isbn: 978-1-4673-7346-3. doi: 10.1109/ICAMIMIA.2015.7508033.

[215] Paden, B., Cap, M., Yong, S. Z., et al. *A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles*. 2016. url: http://arxiv.org/pdf/1604.07446v1.

[216] Xidias, E., Zacharia, P., and Nearchou, A. "Path Planning and scheduling for a fleet of autonomous vehicles". In: *Robotica* 34.10 (2016), pp. 2257–2273. issn: 0263-5747. doi: 10.1017/S0263574714002872.

[217] International Federation of Robotics. *Executive Summary World Robotics 2018 Industrial Robots*. 2018. url: https://ifr.org/downloads/press2018/Executive_Summary_WR_2018_Industrial_Robots.pdf (visited on 2018-10-26).

[218] Gecks, T. and Henrich, D. "SIMERO: Camera Supervised Workspace for Service Robots". In: *2nd Workshop on Advances in Service Robotics, Fraunhofer IPA*. 2004.

[219] Lenz, C., Grimm, M., R?oder, T., et al. *Fusing multiple Kinects to survey shared Human-Robot-Workspaces*. 2012.

[220] Döttling, D. *SafetyEYE: Innovatives Kamerasystem zur Überwachung von Gefahrbereichen*. Stuttgart, 2014. url: https://www.hft-stuttgart.de/Studienbereiche/Vermessung/Bachelor-Informationslogistik/Aktuell/Veranstaltungen/inflogtag2014/SafetyEYE_HFT-Stuttgart_09-04-14_genehmigt.pdf (visited on 2018-10-27).

[221] Bascetta, L., Ferretti, G., Rocco, P., et al. "Towards safe human-robot interaction in robotic cells: An approach based on visual tracking and intention estimation". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 2971–2978. isbn: 978-1-61284-456-5. doi: 10.1109/IROS.2011.6094642.

[222] Lasota, P. A., Fong, T., and Shah, J. A. "A Survey of Methods for Safe Human-Robot Interaction". In: *Foundations and Trends in Robotics* 5.3 (2014), pp. 261–349. issn: 1935-8253. doi: 10.1561/2300000052.

[223] Bauer, Andrea, Wollherr, et al. "Human-Robot Collaboration: A SURVEY". In: *International Jour-*

nal of Humanoid Robotics 05.01 (2008), pp. 47–66. ISSN: 0219-8436. DOI: 10.1142/S0219843608001303.

[224] Ajoudani, A., Zanchettin, A. M., Ivaldi, S., et al. “Progress and prospects of the human–robot collaboration”. In: *Autonomous Robots* 42.5 (2018), pp. 957–975. ISSN: 0929-5593. DOI: 10.1007/s10514-017-9677-2.

[225] Zafari, F., Gkelias, A., and Leung, K. *A Survey of Indoor Localization Systems and Technologies*. 2018. URL: http://arxiv.org/pdf/1709.01015v2.

[226] Brena, R. F., García-Vázquez, J. P., Galván-Tejada, C. E., et al. “Evolution of Indoor Positioning Technologies: A Survey”. In: *Journal of Sensors* 2017.6, article 359 (2017), pp. 1–21. DOI: 10.1155/2017/2630413.

[227] Sakpere, W., Adeyeye Oshin, M., and Mlitwa, N. B. W. “A State-of-the-Art Survey of Indoor Positioning and Navigation Systems and Technologies”. In: *South African Computer Journal* 29.3 (2017). ISSN: 2313-7835. DOI: 10.18489/sacj.v29i3.452.

[228] Mainetti, L., Patrono, L., and Sergi, I. “A survey on indoor positioning systems”. In: *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2014, pp. 111–120. ISBN: 978-9-5329-0052-1. DOI: 10.1109/SOFTCOM.2014.7039067.

[229] Huang, G. Q., Zhang, Y. F., and Jiang, P. Y. “RFID-based wireless manufacturing for real-time management of job shop WIP inventories”. In: *The International Journal of Advanced Manufacturing Technology* 36.7-8 (2008), pp. 752–764. ISSN: 0268-3768. DOI: 10.1007/s00170-006-0897-4.

[230] Meyer, G. G., Främling, K., and Holmström, J. “Intelligent Products: A survey”. In: *Computers in Industry* 60.3 (2009), pp. 137–148. ISSN: 01663615. DOI: 10.1016/j.compind.2008.12.005.

[231] McFarlane, D., Sarma, S., Chirn, J. L., et al. “The Intelligent Product in Manufacturing Control and Management”. In: *IFAC Proceedings Volumes* 35.1 (2002), pp. 49–54. ISSN: 14746670. DOI: 10.3182/20020721-6-ES-1901.00011.

[232] Khoo, B. K. “RFID Technology in the Supply Chain: Issues, and Implications for Research and Practice”. In: *Annual Decision Science Institute Meeting & Conference*. 2013.

[233] Zhang, Y., Jiang, P., Huang, G., et al. “RFID-enabled real-time manufacturing information tracking infrastructure for extended enterprises”. In: *Journal of Intelligent Manufacturing* 23.6 (2012), pp. 2357–2366. ISSN: 0956-5515. DOI: 10.1007/s10845-010-0475-3.

[234] Baygin, M., Karakose, M., Sarimaden, A., et al. *An Image Processing based Object Counting Approach for Machine Vision Application*. 2018. URL: http://arxiv.org/pdf/1802.05911v1.

[235] Malamas, E. N., Petrakis, E. G. M., Zervakis, M., et al. *A Survey on Industrial Vision Systems , Applications and Tools 1*. 2002.

[236] Brosnan, T. and Sun, D.-W. “Improving quality inspection of food products by computer vision—-a review”. In: *Journal of Food Engineering* 61.1 (2004), pp. 3–16. ISSN: 02608774. DOI: 10.1016/S0260-8774(03)00183-3.

[237] Saldaña, E., Siche, R., Luján, M., et al. “Review: Computer vision applied to the inspection and quality control of fruits and vegetables”. In: *Brazilian Journal of Food Technology* 16.4 (2013), pp. 254–272. ISSN: 1981-6723. DOI: 10.1590/S1981-67232013005000031.

[238] Gunasekaran, S. “Computer vision technology for food quality assurance”. In: *Trends in Food Science & Technology* 7.8 (1996), pp. 245–256. ISSN: 09242244. DOI: 10.1016/0924-2244(96)10028-5.

[239] Anagnostopoulos, C., Vergados, D., Kayafas, E., et al. “A computer vision approach for textile quality control”. In: *The Journal of Visualization and Computer Animation* 12.1 (2001), pp. 31–44. ISSN: 1049-8907. DOI: 10.1002/vis.245.

[240] Kumar, A. “Computer-Vision-Based Fabric Defect Detection: A Survey”. In: *IEEE Transactions on Industrial Electronics* 55.1 (2008), pp. 348–363. ISSN: 0278-0046. DOI: 10.1109/TIE.1930.896476.

[241] Huang, S.-H. and Pan, Y.-C. “Automated visual inspection in the semiconductor industry: A survey”. In: *Computers in Industry* 66 (2015), pp. 1–10. ISSN: 01663615. DOI: 10.1016/j.compind.2014.10.006.

[242] Tout, K. “Automatic vision system for surface inspection and monitoring: Application to wheel inspection”. Theses. Université de Technologie de Troyes - UTT, 2018. URL: https://tel.archives-ouvertes.fr/tel-01801803.

[243] Liu, Y., Li, S., Wang, J., et al. “A computer vision-based assistant system for the assembly of narrow cabin products”. In: *The International Journal of Advanced Manufacturing Technology* 76.1-4 (2015), pp. 281–293. ISSN: 0268-3768. DOI: 10.1007/s00170-014-6274-9.

[244] He, K., Zhang, Q., and Hong, Y. “Profile monitoring based quality control method for fused deposition modeling process”. In: *Journal of Intelligent Manufacturing* 25.6 (2018), p. 1349. ISSN: 0956-5515. DOI: 10.1007/s10845-018-1424-9.

[245] Silveira, J., Ferreira, M., Santos, C., et al. “Computer Vision Techniques Applied to the Quality Control of Ceramic Plates”. In: (2009).

[246] Kottari, K., Delibasis, K., and Plagianakos, V. “Real time vision-based measurements for quality control of industrial rods on a moving conveyor”. In: *Multimedia Tools and Applications* 77.8 (2018), pp. 9307–9324. ISSN: 1380-7501. DOI: 10.1007/s11042-017-4891-7.

[247] Tarallo, A., Mozzillo, R., Di Gironimo, G., et al. “A cyber-physical system for production monitoring of manual manufacturing processes”. In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 12.4 (2018), pp. 1235–1241. ISSN: 1955-2513. DOI: 10.1007/s12008-018-0493-5.

[248] Hohman, F., Kahng, M., Pienta, R., et al. *Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers*. 2018. URL: http://arxiv.org/pdf/1801.06889v3.

[249] Lee, J., Kao, H.-A., and Yang, S. “Service Innovation and Smart Analytics for Industry 4.0 and Big

Data Environment". In: *Procedia CIRP* 16 (2014), pp. 3–8. ISSN: 22128271. DOI: 10.1016/j.procir.2014.02.001.

[250] BANKER, BARDHAN, CHANG, et al. "Plant Information Systems, Manufacturing Capabilities, and Plant Performance". In: *MIS Quarterly* 30.2 (2006), p. 315. ISSN: 02767783. DOI: 10.2307/25148733.

[251] HOSSEINI-NASAB, H., FEREIDOUNI, S., FATEMI GHOMI, S. M. T., et al. "Classification of facility layout problems: A review study". In: *The International Journal of Advanced Manufacturing Technology* 94.1-4 (2018), pp. 957–977. ISSN: 0268-3768. DOI: 10.1007/s00170-017-0895-8.

[252] DRIRA, A., PIERREVAL, H., and HAJRI-GABOUJ, S. "Facility layout problems: A survey". In: *Annual Reviews in Control* 31.2 (2007), pp. 255–267. ISSN: 13675788. DOI: 10.1016/j.arcontrol.2007.04.001.

[253] DRIRA, A., PIERREVAL, H., and HAJRI-GABOUJ, S. "Facility Laout Problems: A Literature Analysis". In: *IFAC Proceedings Volumes* 39.3 (2006), pp. 389–400. ISSN: 14746670. DOI: 10.3182/20060517-3-FR-2903.00208.

[254] SINGH, S. P. and SHARMA, R. R. K. "A review of different approaches to the facility layout problems". In: *The International Journal of Advanced Manufacturing Technology* 30.5-6 (2006), pp. 425–433. ISSN: 0268-3768. DOI: 10.1007/s00170-005-0087-9.

[255] SHOUMAN, M. A., NAWARA, G., REYAD, A. H., et al. *Problem (FLP) and Intelligent Techniques: A Survey.* 2000.

[256] NIEBLES, F., ESCOBAR, I., AGUDELO, L., et al. "A Comparative Analysis of Genetic Algorithms and QAP Formulation for Facility Layout Problem: An Application in a Real Context". In: *Advances in swarm intelligence*. Ed. by TAN, Y., SHI, Y., and LI, L. Lecture notes in computer science Theoretical computer science and general issues. Cham and Heidelberg: Springer, 2016, pp. 59–75. ISBN: 978-3-319-41009-8.

[257] WAGNER, U., ALGEDDAWY, T., ELMARAGHY, H., et al. "Product Family Design for Changeable Learning Factories". In: *Procedia CIRP* 17 (2014), pp. 195–200. ISSN: 22128271. DOI: 10.1016/j.procir.2014.01.119.

[258] REITZE, A., JÜRGENSMEYER, N., LIER, S., et al. "Roadmap for a Smart Factory: A Modular, Intelligent Concept for the Production of Specialty Chemicals". In: *Angewandte Chemie (International ed. in English)* 57.16 (2018), pp. 4242–4247. DOI: 10.1002/anie.201711571.

[259] YUSOF, Y. and LATIF, K. "Survey on computer-aided process planning". In: *The International Journal of Advanced Manufacturing Technology* 75.1-4 (2014), pp. 77–89. ISSN: 0268-3768. DOI: 10.1007/s00170-014-6073-3.

[260] BHARATH V. G. and PATIL, R. "Virtual Manufacturing: A Review". In: *International Journal of Engineering Research & Technology (IJERT)*. 2015, pp. 355–364.

[261] SOUZA, M. C. F., SACCO, M., and PORTO, A. J. V. "Virtual manufacturing as a way for the factory of the future". In: *Journal of Intelligent Manufacturing* 17.6 (2006), pp. 725–735. ISSN: 0956-5515. DOI: 10.1007/s10845-006-0041-1.

[262] AL-AHMARI, A. M., ABIDI, M. H., AHMAD, A., et al. "Development of a virtual manufacturing assembly simulation system". In: *Advances in Mechanical Engineering* 8.3 (2016), p. 168781401663982. ISSN: 1687-8140. DOI: 10.1177/1687814016639824.

[263] TERKAJ, W., TOLIO, T., and URGO, M. "A virtual factory approach for in situ simulation to support production and maintenance planning". In: *CIRP Annals* 64.1 (2015), pp. 451–454. ISSN: 00078506. DOI: 10.1016/j.cirp.2015.04.121.

[264] TERKAJ, W. and URGO, M. "A Virtual Factory Data Model as a Support Tool for the Simulation of Manufacturing Systems". In: *Procedia CIRP* 28 (2015), pp. 137–142. ISSN: 22128271. DOI: 10.1016/j.procir.2015.04.023.

[265] TOLIO, T., SACCO, M., TERKAJ, W., et al. "Virtual Factory: An Integrated Framework for Manufacturing Systems Design and Analysis". In: *Procedia CIRP* 7 (2013), pp. 25–30. ISSN: 22128271. DOI: 10.1016/j.procir.2013.05.005.

[266] KÁDÁR, B., TERKAJ, W., and SACCO, M. "Semantic Virtual Factory supporting interoperable modelling and evaluation of production systems". In: *CIRP Annals* 62.1 (2013), pp. 443–446. ISSN: 00078506. DOI: 10.1016/j.cirp.2013.03.045.

[267] BACK, M., CHILDS, T., DUNNIGAN, A., et al. "The virtual factory: Exploring 3D worlds as industrial collaboration and control environments". In: *2010 IEEE Virtual Reality Conference (VR)*. IEEE, 2010, pp. 257–258. ISBN: 978-1-4244-6237-7. DOI: 10.1109/VR.2010.5444777.

[268] LIN, M.-H. and FU, L.-C. "A virtual factory based approach to on-line simulation and scheduling for an FMS and a case study". In: *Journal of Intelligent Manufacturing* 12.3 (2001), pp. 269–279. ISSN: 0956-5515. DOI: 10.1023/A:1011201009821.

[269] KUNZ, A., ZANK, M., FJELD, M., et al. "Real Walking in Virtual Environments for Factory Planning and Evaluation". In: *Procedia CIRP* 44 (2016), pp. 257–262. ISSN: 22128271. DOI: 10.1016/j.procir.2016.02.086.

[270] UHLEMANN, T. H.-J., LEHMANN, C., and STEINHILPER, R. "The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0". In: *Procedia CIRP* 61 (2017), pp. 335–340. ISSN: 22128271. DOI: 10.1016/j.procir.2016.11.152.

[271] KÁDÁR, B., LENGYEL, A., MONOSTORI, L., et al. "Enhanced control of complex production structures by tight coupling of the digital and the physical worlds". In: *CIRP Annals* 59.1 (2010), pp. 437–440. ISSN: 00078506. DOI: 10.1016/j.cirp.2010.03.123.

[272] CHANDOLA, V., BANERJEE, A., and KUMAR, V. "Anomaly detection: A Survey". In: *ACM Computing Surveys* 41.3 (2009), pp. 1–58. ISSN: 03600300. DOI: 10.1145/1541880.1541882.

[273] XIAO, T., ZHANG, C., and ZHA, H. "Learning to Detect Anomalies in Surveillance Video". In: *IEEE*

Signal Processing Letters 22.9 (2015), pp. 1477–1481. ISSN: 1070-9908. DOI: 10.1109/LSP.2015.2410031.

[274] SAINI, D. K., AHIR, D., and GANATRA, A. "Techniques and Challenges in Building Intelligent Systems: Anomaly Detection in Camera Surveillance". In: *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 2*. Ed. by SATAPATHY, S. C. and DAS, S. Vol. 51. Smart Innovation, Systems and Technologies. Cham: Springer International Publishing, 2016, pp. 11–21. ISBN: 978-3-319-30926-2. DOI: 10.1007/978-3-319-30927-9_2.

[275] SACCO, M., REDAELLI, C., CANDEA, C., et al. "DiFac: An integrated scenario for the Digital Factory". In: *2009 IEEE International Technology Management Conference (ICE)*. IEEE, 2009, pp. 1–8. ISBN: 978-0-85358-259-5. DOI: 10.1109/ITMC.2009.7461380.

[276] PATLE, D. S., MANCA, D., NAZIR, S., et al. "Operator training simulators in virtual reality environment for process operators: A review". In: *Virtual Reality* 31.1 (2018), p. 588. ISSN: 1359-4338. DOI: 10.1007/s10055-018-0354-3.

[277] BLUEMEL, E., HINTZE, A., SCHULZ, T., et al. "Virtual environments for the training of maintenance and service tasks". In: *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*. IEEE, 2003, pp. 2001–2007. ISBN: 0-7803-8131-9. DOI: 10.1109/WSC.2003.1261664.

[278] SCHENK, M., STRASSBURGER, S., and KISSNER, H. "Combining virtual reality and assembly simulation for production planning and worker qualification". In: *1st International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2005)*. Ed. by ZÄH, M. and REINHART, G. München: Utz, 2005, pp. 411–414. ISBN: 3-8316-0540-8.

[279] GORECKY, D., KHAMIS, M., and MURA, K. "Introduction and establishment of virtual training in the factory of the future". In: *International Journal of Computer Integrated Manufacturing* 2.2 (2015), pp. 1–9. DOI: 10.1080/0951192X.2015.1067918.

[280] MENCK, N., YANG, X., WEIDIG, C., et al. "Collaborative Factory Planning in Virtual Reality". In: *Procedia CIRP* 3 (2012), pp. 317–322. ISSN: 22128271. DOI: 10.1016/j.procir.2012.07.055.

[281] ZINNIKUS, I., CAO, X., KLUSCH, M., et al. "A Collaborative Virtual Workspace for Factory Configuration and Evaluation". In: *Proceedings of the 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Ed. by BERTINO, E., GEORGAKOPOULOS, D., SRIVATSA, M., et al. ICST, 2013. ISBN: 978-1-936968-92-3. DOI: 10.4108/icst.collaboratecom.2013.254053.

[282] SACCO, M., PEDRAZZOLI, P., and TERKAJ, W. "VFF: Virtual Factory Framework". In: *2010 IEEE International Technology Management Conference (ICE)*. IEEE, 2010, pp. 1–8. ISBN: 978-1-62748-686-6. DOI: 10.1109/ICE.2010.7477041.

[283] BAQERSAD, J., POOZESH, P., NIEZRECKI, C., et al. "Photogrammetry and optical methods in structural dynamics – A review". In: *Mechanical Systems and Signal Processing* 86 (2017), pp. 17–34. ISSN: 08883270. DOI: 10.1016/j.ymssp.2016.02.011.

[284] ZHANG, D., GUO, J., LEI, X., et al. "A High-Speed Vision-Based Sensor for Dynamic Vibration Analysis Using Fast Motion Extraction Algorithms". In: *Sensors (Basel, Switzerland)* 16.4 (2016). DOI: 10.3390/s16040572.

[285] YAVARI, E., NUTI, P., and BORIC-LUBECKE, O. "Occupancy detection using radar noise floor". In: *2016 IEEE/ACES International Conference on Wireless Information Technology and Systems (ICWITS) and Applied Computational Electromagnetics (ACES)*. IEEE, 2016, pp. 1–3. ISBN: 978-1-5090-1259-6. DOI: 10.1109/ROPACES.2016.7465363.

[286] KRIZHEVSKY, A., SUTSKEVER, I., and HINTON, G. E. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2012), pp. 84–90. ISSN: 00010782. DOI: 10.1145/3065386.

[287] ZEILER, M. D. and FERGUS, R. *Visualizing and Understanding Convolutional Networks*. 2013. URL: http://arxiv.org/pdf/1311.2901v3.

[288] SIMONYAN, K. and ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. URL: http://arxiv.org/pdf/1409.1556v6.

[289] SZEGEDY, C., LIU, W., JIA, Y., et al. *Going Deeper with Convolutions*. 2014. URL: http://arxiv.org/pdf/1409.4842v1.

[290] IOFFE, S. and SZEGEDY, C. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. URL: http://arxiv.org/pdf/1502.03167v3.

[291] SZEGEDY, C., VANHOUCKE, V., IOFFE, S., et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. URL: http://arxiv.org/pdf/1512.00567v3.

[292] HOWARD, A. G., ZHU, M., CHEN, B., et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. URL: http://arxiv.org/pdf/1704.04861v1.

[293] HUANG, G., LIU, Z., VAN DER MAATEN, L., et al. *Densely Connected Convolutional Networks*. 2016. URL: http://arxiv.org/pdf/1608.06993v5.

[294] XIE, S., GIRSHICK, R., DOLLÁR, P., et al. *Aggregated Residual Transformations for Deep Neural Networks*. 2016. URL: http://arxiv.org/pdf/1611.05431v2.

[295] DENG, J., DONG, W., SOCHER, R., et al. "ImageNet: A large-scale hierarchical image database". In: *CVPR 2009*. Los Alamitos, California: IEEE, 2009, pp. 248–255. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206848.

[296] KRIZHEVSKY, A. *Learning Multiple Layers of Features from Tiny Images*. 2009.

[297] REDMON, J., DIVVALA, S., GIRSHICK, R., et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2015. URL: http://arxiv.org/pdf/1506.02640v5.

[298] REDMON, J. and FARHADI, A. *YOLO9000: Better, Faster, Stronger*. 2016. URL: http://arxiv.org/pdf/1612.08242v1.

[299] Redmon, J. *YOLO Variants.* 2018. url: https://pjreddie.com/darknet/imagenet/#extraction (visited on 2018-08-17).

[300] Pinto, N., Cox, D. D., and DiCarlo, J. J. “Why is real-world visual object recognition hard?” In: *PLoS computational biology* 4.1 (2008), e27. doi: 10.1371/journal.pcbi.0040027.

[301] Ponce, J., Berg, T. L., Everingham, M., et al. “Dataset Issues in Object Recognition”. In: *Toward category-level object recognition.* Ed. by Ponce, J. Vol. 4170. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 29–48. isbn: 978-3-540-68794-8. doi: 10.1007/11957959_2.

[302] Lin, T.-Y., Maire, M., Belongie, S., et al. *Microsoft COCO: Common Objects in Context.* 2014. url: http://arxiv.org/pdf/1405.0312v3.

[303] Everingham, M., van Gool, L., Williams, C. K. I., et al. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338. issn: 0920-5691. doi: 10.1007/s11263-009-0275-4.

[304] Everingham, M., Eslami, S. M. A., van Gool, L., et al. “The Pascal Visual Object Classes Challenge: A Retrospective”. In: *International Journal of Computer Vision* 111.1 (2015), pp. 98–136. issn: 0920-5691. doi: 10.1007/s11263-014-0733-5.

[305] Russakovsky, O., Deng, J., Su, H., et al. *ImageNet Large Scale Visual Recognition Challenge.* 2015. url: http://arxiv.org/pdf/1409.0575v3.

[306] Krasin, I., Duerig, T., Alldrin, N., et al. *OpenImages: A public dataset for large-scale multi-label and multi-class image classification.* 2017. url: https://storage.googleapis.com/openimages/web/index.html.

[307] Dollar, P., Wojek, C., Schiele, B., et al. “Pedestrian detection: A benchmark”. In: *CVPR 2009.* Los Alamitos, California: IEEE, 2009, pp. 304–311. isbn: 978-1-4244-3992-8. doi: 10.1109/CVPR.2009.5206631.

[308] Dollár, P., Wojek, C., Schiele, B., et al. “Pedestrian detection: An evaluation of the state of the art”. In: *IEEE transactions on pattern analysis and machine intelligence* 34.4 (2012), pp. 743–761. issn: 0162-8828. doi: 10.1109/TPAMI.2011.155.

[309] Fei-Fei, L., Fergus, R., and Perona, P. “One-shot learning of object categories”. In: *IEEE transactions on pattern analysis and machine intelligence* 28.4 (2006), pp. 594–611. issn: 0162-8828. doi: 10.1109/TPAMI.2006.79.

[310] Griffin, G., Holub, A., and Perona, P. *Caltech-256 Object Category Dataset.* 2007.

[311] Taylor, L. and Nitschke, G. *Improving Deep Learning using Generic Data Augmentation.* 2017. url: http://arxiv.org/pdf/1708.06020v1.

[312] Perez, L. and Wang, J. *The Effectiveness of Data Augmentation in Image Classification using Deep Learning.* 2017. url: http://arxiv.org/pdf/1712.04621v1.

[313] Rahman, M. A. and Wang, Y. “Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation”. In: *ISVC* (2016).

[314] Sermanet, P., Eigen, D., Zhang, X., et al. *OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks.* 2013. url: http://arxiv.org/pdf/1312.6229v4.

[315] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to information retrieval.* Reprinted. Cambridge: Cambridge Univ. Press, 2009. isbn: 9780521865715.

[316] Cartucho, J. *Explanation of mean Average Precision: GitHub repository.* 2018. url: https://github.com/Cartucho/mAP (visited on 2018-08-17).

[317] Altenberger, F. and Lenz, C. *A Non-Technical Survey on Deep Convolutional Neural Network Architectures.* 2018. url: http://arxiv.org/pdf/1803.02129v1.

[318] Rey, J. *Object Detection: A Guide in the Age of Deep Learning.* 2017. url: https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/ (visited on 2018-08-17).

[319] Ouaknine, A. *Review of Deep Learning Algorithms for Object Detection.* 2018. url: https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852 (visited on 2018-08-17).

[320] Xu, J. *Deep Learning for Object Detection: A Comprehensive Review.* 2017. url: https://www.kdnuggets.com/2017/10/deep-learning-object-detection-comprehensive-review.html (visited on 2018-08-17).

[321] Hui, J. *Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3.* 2018. url: https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088 (visited on 2018-08-17).

[322] Forson, E. *Understanding SSD MultiBox: ?Real-Time Object Detection In Deep Learning.* 2017. url: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab (visited on 2018-08-17).

[323] Hui, J. *Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3).* 2018. url: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359 (visited on 2018-08-17).

[324] Chablani, M. *Overfeat paper: ?Summary.* 2017. url: https://medium.com/@ManishChablani/overfeat-paper-summary-b55060eeb991 (visited on 2018-08-17).

[325] He, K., Zhang, X., Ren, S., et al. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”. In: *(Keine Angabe)* 8691 (2014), pp. 346–361. issn: 0302-9743. doi: 10.1007/978-3-319-10578-9_23. url: http://arxiv.org/pdf/1406.4729v4.

[326] Girshick, R. *Fast R-CNN.* 2015. url: http://arxiv.org/pdf/1504.08083v2.

[327] Lin, T.-Y., Dollár, P., Girshick, R., et al. *Feature Pyramid Networks for Object Detection.* 2016. url: http://arxiv.org/pdf/1612.03144v2.

[328] Prakash, J. *The intuition behind RetinaNet.* 2018. URL: https://medium.com/@14prakash/the-intuition-behind-retinanet-eb636755607d (visited on 2018-08-17).

[329] Szegedy, C., Toshev, A., and Erhan, D. *Deep Neural Networks for Object Detection.* NIPS. 2013.

[330] Erhan, D., Szegedy, C., Toshev, A., et al. *Scalable Object Detection using Deep Neural Networks.* 2013. URL: http://arxiv.org/pdf/1312.2249v1.

[331] Ren, S., He, K., Girshick, R., et al. *Object Detection Networks on Convolutional Feature Maps.* 2016. URL: http://arxiv.org/pdf/1504.06066v2.

[332] Gidaris, S. and Komodakis, N. *Object detection via a multi-region & semantic segmentation-aware CNN model.* 2015. URL: http://arxiv.org/pdf/1505.01749v3.

[333] Lenc, K. and Vedaldi, A. *R-CNN minus R.* 2015. URL: http://arxiv.org/pdf/1506.06981v1.

[334] Szegedy, C., Reed, S., Erhan, D., et al. *Scalable, High-Quality Object Detection.* 2015. URL: http://arxiv.org/pdf/1412.1441v3.

[335] Bell, S., Zitnick, C. L., Bala, K., et al. *Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks.* 2015. URL: http://arxiv.org/pdf/1512.04143v1.

[336] Kong, T., Yao, A., Chen, Y., et al. *HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection.* 2016. URL: http://arxiv.org/pdf/1604.00600v1.

[337] Cai, Z., Fan, Q., Feris, R. S., et al. *A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection.* 2016. URL: http://arxiv.org/pdf/1607.07155v1.

[338] Fu, C.-Y., Liu, W., Ranga, A., et al. *DSSD: Deconvolutional Single Shot Detector.* 2017. URL: http://arxiv.org/pdf/1701.06659v1.

[339] Leal-Taixé, L., Milan, A., Reid, I., et al. *MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking.* 2015. URL: http://arxiv.org/pdf/1504.01942v1.

[340] Li, X., Hu, W., Shen, C., et al. *A Survey of Appearance Models in Visual Object Tracking.* 2013. URL: http://arxiv.org/pdf/1303.4803v1.

[341] Shaikh, S. H., Saeed, K., and Chaki, N. "Moving Object Detection Approaches, Challenges and Object Tracking". In: *Moving Object Detection Using Background Subtraction.* Ed. by Shaikh, S. H., Saeed, K., and Chaki, N. SpringerBriefs in Computer Science. Cham: Springer International Publishing, 2014, pp. 5–14. ISBN: 978-3-319-07385-9. DOI: 10.1007/978-3-319-07386-6_2.

[342] Doi, M., Matsumoto, T., Kimachi, A., et al. "Robust color object tracking method against illumination color Change". In: *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS).* IEEE, 2014, pp. 718–722. ISBN: 978-1-4799-5955-6. DOI: 10.1109/SCIS-ISIS.2014.7044769.

[343] Zang, D. "Illumination invariant object tracking based on multiscale phase". In: *2010 IEEE International Conference on Image Processing.* IEEE, 2010, pp. 357–360. ISBN: 978-1-4244-7992-4. DOI: 10.1109/ICIP.2010.5651716.

[344] Rubio, J. C., Serrat, J., and López, A. M. "Multiple Target Tracking and Identity Linking under Split, Merge and Occlusion of Targets and Observations". In: *ICPRAM* (2012).

[345] Storlie, C. B., Lee, C. T., Hannig, J., et al. "Tracking of multiple merging and splitting targets: A statistical perspective". In: *Statistica Sinica* 19 (2009).

[346] Fan, R., Zhang, F.-L., Zhang, M., et al. "Robust tracking-by-detection using a selection and completion mechanism". In: *Computational Visual Media* 3.3 (2017), pp. 285–294. ISSN: 2096-0433. DOI: 10.1007/s41095-017-0083-7.

[347] Jiang, N., Liu, W., Su, H., et al. "Tracking low resolution objects by metric preservation". In: *CVPR 2011.* IEEE, 2011, pp. 1329–1336. ISBN: 978-1-4577-0394-2. DOI: 10.1109/CVPR.2011.5995537.

[348] Bao, X., Zhan, H., and Yuan, J. "Study of target tracking and recovery with low resolution image series". In: *2017 3rd IEEE International Conference on Computer and Communications (ICCC).* IEEE, 2017, pp. 1836–1840. ISBN: 978-1-5090-6352-9. DOI: 10.1109/CompComm.2017.8322856.

[349] Kaew Tra Kul Pong, P. and Bowden, R. "Adaptive Visual System for Tracking Low Resolution Colour Targets". In: *Procedings of the British Machine Vision Conference 2001.* Ed. by Cootes, T. and Taylor, C. British Machine Vision Association, 2001, pp. 26.1–26.10. ISBN: 1-901725-16-2. DOI: 10.5244/C.15.26.

[350] Tang, S., Andriluka, M., Milan, A., et al. "Learning People Detectors for Tracking in Crowded Scenes". In: *2013 IEEE International Conference on Computer Vision.* IEEE, 2013, pp. 1049–1056. ISBN: 978-1-4799-2840-8. DOI: 10.1109/ICCV.2013.134.

[351] Rahmatian, S. and Safabakhsh, R. "Online multiple people tracking-by-detection in crowded scenes". In: *7'th International Symposium on Telecommunications (IST'2014).* IEEE, 2014, pp. 337–342. ISBN: 978-1-4799-5359-2. DOI: 10.1109/ISTEL.2014.7000725.

[352] Ramanan, D., Forsyth, D. A., and Zisserman, A. "Tracking People by Learning Their Appearance". In: *IEEE transactions on pattern analysis and machine intelligence* 29.1 (2007), pp. 65–81. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.250600.

[353] Liu, W., Camps, O., and Sznaier, M. *Multi-camera Multi-Object Tracking.* 2017. URL: http://arxiv.org/pdf/1709.07065v1.

[354] Wang, Y., Lu, K., and Zhai, R. "Technique and Challange for Multi-Camera Tracking". In: *2014 7th International Congress on Image and Signal Processing.* IEEE, 2014, pp. 32–37. ISBN: 978-1-4799-5835-1. DOI: 10.1109/CISP.2014.7003745.

[355] Zhang, Z., Wu, J., Zhang, X., et al. *Multi-Target, Multi-Camera Tracking by Hierarchical Clustering: Recent Progress on DukeMTMC Project.* 2017. URL: http://arxiv.org/pdf/1712.09531v1.

[356] Chavdarova, T. and Fleuret, F. *Deep Multi-camera People Detection.* 2017. URL: http://arxiv.org/pdf/1702.04593v3.

[357] Silveira, G. and Malis, E. "Real-time Visual Tracking under Arbitrary Illumination Changes". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–6. isbn: 1-4244-1179-3. doi: 10.1109/CVPR.2007.382993.

[358] Hu, W., Li, X., Zhang, X., et al. "Incremental Tensor Subspace Learning and Its Applications to Foreground Segmentation and Tracking". In: *International Journal of Computer Vision* 91.3 (2011), pp. 303–327. issn: 0920-5691. doi: 10.1007/s11263-010-0399-6.

[359] Mason, M. and Duric, Z. "Using histograms to detect and track objects in color video". In: *Proceedings 30th Applied Imagery Pattern Recognition Workshop (AIPR 2001). Analysis and Understanding of Time Varying Imagery*. IEEE Comput. Soc, 2001, pp. 154–159. isbn: 0-7695-1245-3. doi: 10.1109/AIPR.2001.991219.

[360] Xiao, C., Chen, W., and Gao, H. "Object tracking algorithm based on HSV color histogram and block-sparse representation". In: *2015 34th Chinese Control Conference (CCC)*. IEEE, 2015, pp. 3826–3831. isbn: 978-9-8815-6389-7. doi: 10.1109/ChiCC.2015.7260229.

[361] Dash, P. P., Patra, D., Mishra, S. K., et al. *Kernel based Object Tracking using Color Histogram Technique*. 2012.

[362] Maggio, E. and Cavallaro, A. "Multi-part target representation for color tracking". In: *IEEE International Conference on Image Processing 2005*. IEEE, 2005, pp. I–729. isbn: 0-7803-9134-9. doi: 10.1109/ICIP.2005.1529854.

[363] Birchfield, S. "Elliptical head tracking using intensity gradients and color histograms". In: *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*. IEEE Comput. Soc, 1998, pp. 232–237. isbn: 0-8186-8497-6. doi: 10.1109/CVPR.1998.698614.

[364] Liu, T.-L. and Chen, H.-T. "Real-time tracking using trust-region methods". In: *IEEE transactions on pattern analysis and machine intelligence* 26.3 (2004), pp. 397–402. issn: 0162-8828. doi: 10.1109/TPAMI.2004.1262335.

[365] Maggio, E., Smerladi, F., and Cavallaro, A. "Adaptive Multifeature Tracking in a Particle Filtering Framework". In: *IEEE Transactions on Circuits and Systems for Video Technology* 17.10 (2007), pp. 1348–1359. issn: 1051-8215. doi: 10.1109/TCSVT.2007.903781.

[366] Maggio, E., Smeraldi, F., and Cavallaro, A. "Combining Colour and Orientation for Adaptive Particle Filter–based Tracking". In: *Procedings of the British Machine Vision Conference 2005*. Ed. by Clocksin, W. F., Fitzgibbon, A. W., and Torr, P. H. S. British Machine Vision Association, 2005, pp. 79.1–79.10. isbn: 1-901725-29-4. doi: 10.5244/C.19.79.

[367] Bilinski, P., Bremond, F., and Kaaniche, M. B. "Multiple object tracking with occlusions using HOG descriptors and multi resolution images". In: *3rd International Conference on Imaging for Crime Detection and Prevention (ICDP 2009)*. IET, 2009, P36–P36. doi: 10.1049/ic.2009.0264.

[368] Wang, Y., Jiang, L., Liu, Q., et al. "Optimal Appearance Model for Visual Tracking". In: *PloS one* 11.1 (2016), e0146763. doi: 10.1371/journal.pone.0146763.

[369] Zhou, S. K., Chellappa, R., and Moghaddam, B. "Visual Tracking and Recognition Using Appearance-Adaptive Models in Particle Filters". In: *IEEE Transactions on Image Processing* 13.11 (2004), pp. 1491–1506. issn: 1057-7149. doi: 10.1109/TIP.2004.836152.

[370] Lucas, B. D. and Kanade, T. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence*. IJCAI '81. 1981, pp. 674–679.

[371] Wu, Y., Cheng, J., Wang, J., et al. "Real-time visual tracking via Incremental Covariance Tensor Learning". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1631–1638. isbn: 978-1-4244-4420-5. doi: 10.1109/ICCV.2009.5459369.

[372] Wu, Y., Cheng, J., Wang, J., et al. "Real-time probabilistic covariance tracking with efficient model update". In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 21.5 (2012), pp. 2824–2837. doi: 10.1109/TIP.2011.2182521.

[373] Nejhum, S., Rushdi, M., and Ho, J. "Visual Tracking Using Superpixel-Based Appearance Model". In: *Computer Vision Systems*. Ed. by Hutchison, D., Kanade, T., Kittler, J., et al. Vol. 7963. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 213–222. isbn: 978-3-642-39401-0. doi: 10.1007/978-3-642-39402-7_22.

[374] He, C., Zheng, Y. F., and Ahalt, S. C. "Object tracking using the Gabor wavelet transform and the golden section algorithm". In: *IEEE Transactions on Multimedia* 4.4 (2002), pp. 528–538. issn: 1520-9210. doi: 10.1109/TMM.2002.806534.

[375] Tang, F. and Tao, H. "Probabilistic Object Tracking With Dynamic Attributed Relational Feature Graph". In: *IEEE Transactions on Circuits and Systems for Video Technology* 18.8 (2008), pp. 1064–1074. issn: 1051-8215. doi: 10.1109/TCSVT.2008.927106.

[376] Donoser, M. and Bischof, H. "Efficient Maximally Stable Extremal Region (MSER) Tracking". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*. IEEE, 2006, pp. 553–560. isbn: 0-7695-2597-0. doi: 10.1109/CVPR.2006.107.

[377] He, W., Yamashita, T., Lu, H., et al. "SURF Tracking". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1586–1592. isbn: 978-1-4244-4420-5. doi: 10.1109/ICCV.2009.5459360.

[378] Salmane, H., Ruichek, Y., and Khoudour, L. "Object tracking using Harris corner points based optical flow propagation and Kalman filter". In: *2011 14th*

International IEEE Conference on Intelligent Transportation Systems (ITSC). IEEE, 2011, pp. 67–73. ISBN: 978-1-4577-2197-7. DOI: 10.1109/ITSC.2011.6083031.

[379] Yilmaz, A., Javed, O., and Shah, M. "Object tracking: A Survey". In: *ACM Computing Surveys* 38.4 (2006), 13–es. ISSN: 03600300. DOI: 10.1145/1177352.1177355.

[380] Kitagawa, G. "Non-Gaussian State-Space Modeling of Nonstationary Time Series". In: *Journal of the American Statistical Association* 82.400 (1987), p. 1032. ISSN: 0162-1459. DOI: 10.2307/2289375.

[381] Broida, T. J. and Chellappa, R. "Estimation of Object Motion Parameters from Noisy Images". In: *IEEE transactions on pattern analysis and machine intelligence* PAMI-8.1 (1986), pp. 90–99. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767755.

[382] Vermaak, J., Godsill, S. J., and Perez, P. "Monte Carlo filtering for multi-target tracking and data association". In: *IEEE Transactions on Aerospace and Electronic Systems* 41.1 (2005), pp. 309–332. ISSN: 0018-9251. DOI: 10.1109/TAES.2005.1413764.

[383] Comaniciu, D., Ramesh, V., and Meer, P. "Kernel-based object tracking". In: *IEEE transactions on pattern analysis and machine intelligence* 25.5 (2003), pp. 564–577. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2003.1195991.

[384] Tomasi, C. and Kanade, T. "Detection and Tracking of Point Features". In: *International Journal of Computer Vision* (1991).

[385] Isard, M. and Blake, A. "Condensation: Conditional Density Propagation for Visual Tracking". In: *International Journal of Computer Vision* 29.1 (1998), pp. 5–28. ISSN: 0920-5691. DOI: 10.1023/A:1008078328650.

[386] Sato, K. and Aggarwal, J. K. "Temporal spatio-velocity transform and its application to tracking and interaction". In: *Computer Vision and Image Understanding* 96.2 (2004), pp. 100–128. ISSN: 10773142. DOI: 10.1016/j.cviu.2004.02.003.

[387] Briechle, K. and Hanebeck, U. D. "Template matching using fast normalized cross correlation". In: ed. by Casasent, D. P. and Chao, T.-H. SPIE Proceedings. SPIE, 2001, pp. 95–102. DOI: 10.1117/12.421129.

[388] Nguyen, H. T. and Smeulders, A. W. M. "Fast occluded object tracking by a robust appearance filter". In: *IEEE transactions on pattern analysis and machine intelligence* 26.8 (2004), pp. 1099–1104. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.45.

[389] Adam, A., Rivlin, E., and Shimshoni, I. "Robust Fragments-based Tracking using the Integral Histogram". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*. IEEE, 2006, pp. 798–805. ISBN: 0-7695-2597-0. DOI: 10.1109/CVPR.2006.256.

[390] Ross, D. A., Lim, J., Lin, R.-S., et al. "Incremental Learning for Robust Visual Tracking". In: *International Journal of Computer Vision* 77.1-3 (2008), pp. 125–141. ISSN: 0920-5691. DOI: 10.1007/s11263-007-0075-7.

[391] Xie, Y. and Wu, C. "Visual Object Tracking using Particle Filtering with Dual Manifold Models". In: *JOURNAL OF INFORMATION SCIENCE AND ENGINEERING* 31 (2015), pp. 283–296.

[392] Kwon, J. and Lee, K. M. "Tracking by Sampling Trackers". In: *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1195–1202. ISBN: 978-1-4577-1102-2. DOI: 10.1109/ICCV.2011.6126369.

[393] Kwon, J. and Lee, K. M. "Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive Basin Hopping Monte Carlo sampling". In: *CVPR 2009*. Los Alamitos, California: IEEE, 2009, pp. 1208–1215. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206502.

[394] Cehovin, L., Kristan, M., and Leonardis, A. "An adaptive coupled-layer visual model for robust visual tracking". In: *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1363–1370. ISBN: 978-1-4577-1102-2. DOI: 10.1109/ICCV.2011.6126390.

[395] Mei, X. and Ling, H. "Robust visual tracking using L1 minimization". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1436–1443. ISBN: 978-1-4244-4420-5. DOI: 10.1109/ICCV.2009.5459292.

[396] Nguyen, H. T. and Smeulders, A. W. M. "Robust Tracking Using Foreground-Background Texture Discrimination". In: *International Journal of Computer Vision* 69.3 (2006), pp. 277–293. ISSN: 0920-5691. DOI: 10.1007/s11263-006-7067-x.

[397] Kalal, Z., Mikolajczyk, K., and Matas, J. "Tracking-Learning-Detection". In: *IEEE transactions on pattern analysis and machine intelligence* 34.7 (2012), pp. 1409–1422. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2011.239.

[398] Hare, S., Golodetz, S., Saffari, A., et al. "Struck: Structured Output Tracking with Kernels". In: *IEEE transactions on pattern analysis and machine intelligence* 38.10 (2016), pp. 2096–2109. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2015.2509974.

[399] Maggio, E. and Cavallaro, A. *Video tracking: Theory and practice*. 1st publ. Chichester: Wiley, 2011. ISBN: 978-0-470-74964-7.

[400] Babenko, B. "Multiple Instance Learning: Algorithms and Applications". In: (2008).

[401] Babenko, B., Yang, M.-H., and Belongie, S. "Visual tracking with online Multiple Instance Learning". In: *CVPR 2009*. Los Alamitos, California: IEEE, 2009, pp. 983–990. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206737.

[402] Babenko, B., Yang, M.-H., and Belongie, S. "Robust Object Tracking with Online Multiple Instance Learning". In: *IEEE transactions on pattern analysis and machine intelligence* 33.8 (2011), pp. 1619–1632. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.226.

[403] Wang, Z., Wang, L., and Zhang, H. "Patch Based Multiple Instance Learning Algorithm for Object Tracking". In: *Computational intelligence and neuroscience* 2017 (2017), p. 2426475. DOI: 10.1155/2017/2426475.

[404] Lan, X., Yuen, P. C., and Chellappa, R. "Robust MIL-Based Feature Template Learning for Object Tracking". In: *AAAI.* 2017.

[405] Yang, H., Qu, S., and Zheng, Z. "Visual tracking via online discriminative multiple instance metric learning". In: *Multimedia Tools and Applications* 77.4 (2018), pp. 4113–4131. issn: 1380-7501. doi: 10.1007/s11042-017-4498-z.

[406] Xu, C., Tao, W., Meng, Z., et al. "Robust visual tracking via online multiple instance learning with Fisher information". In: *Pattern Recognition* 48.12 (2015), pp. 3917–3926. issn: 00313203. doi: 10.1016/j.patcog.2015.06.004.

[407] Abdechiri, M., Faez, K., and Amindavar, H. "Visual object tracking with online weighted chaotic multiple instance learning". In: *Neurocomputing* 247 (2017), pp. 16–30. issn: 09252312. doi: 10.1016/j.neucom.2017.03.032.

[408] Sharma, V. K. and Mahapatra, K. K. "MIL based visual object tracking with kernel and scale adaptation". In: *Signal Processing: Image Communication* 53 (2017), pp. 51–64. issn: 09235965. doi: 10.1016/j.image.2017.01.007.

[409] Stutz, D., Hermans, A., and Leibe, B. "Superpixels: An Evaluation of the State-of-the-Art". In: *Computer Vision and Image Understanding* 166 (2018), pp. 1–27. issn: 10773142. doi: 10.1016/j.cviu.2017.03.007. url: http://arxiv.org/pdf/1612.01601v3.

[410] Jingjing, L., Ying, C., Cheng, Z., et al. "Tracking Using Superpixel Features". In: *2016 Eighth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA).* IEEE, 2016, pp. 878–881. isbn: 978-1-5090-2312-7. doi: 10.1109/ICMTMA.2016.211.

[411] Wang, S., Lu, H., Yang, F., et al. "Superpixel tracking". In: *2011 International Conference on Computer Vision.* IEEE, 2011, pp. 1323–1330. isbn: 978-1-4577-1102-2. doi: 10.1109/ICCV.2011.6126385.

[412] Wang, J., Liu, W., Xing, W., et al. "Two-level superpixel and feedback based visual object tracking". In: *Neurocomputing* 267 (2017), pp. 581–596. issn: 09252312. doi: 10.1016/j.neucom.2017.06.031.

[413] Wang, L., Lu, H., and Yang, M.-H. "Constrained Superpixel Tracking". In: *IEEE transactions on cybernetics* 48.3 (2018), pp. 1030–1041. doi: 10.1109/TCYB.2017.2675910.

[414] Huang, W., Hu, R., Liang, C., et al. "Structural superpixel descriptor for visual tracking". In: *IJCNN 2017.* Piscataway, NJ: IEEE, 2017, pp. 3146–3152. isbn: 978-1-5090-6182-2. doi: 10.1109/IJCNN.2017.7966248.

[415] Li, A. and Yan, S. "Object Tracking With Only Background Cues". In: *IEEE Transactions on Circuits and Systems for Video Technology* 24.11 (2014), pp. 1911–1919. issn: 1051-8215. doi: 10.1109/TCSVT.2014.2317888.

[416] Gomila, C. and Meyer, F. "Graph-based object tracking". In: *ICIP-2003.* Piscataway, NJ: IEEE, 2003. isbn: 0-7803-7750-8. doi: 10.1109/ICIP.2003.1246611.

[417] Du, D., Qi, H., Li, W., et al. "Online Deformable Object Tracking Based on Structure-Aware Hyper-Graph". In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 25.8 (2016), pp. 3572–3584. doi: 10.1109/TIP.2016.2570556.

[418] Wang, T. and Ling, H. "Gracker: A Graph-Based Planar Object Tracker". In: *IEEE transactions on pattern analysis and machine intelligence* 40.6 (2018), pp. 1494–1501. issn: 0162-8828. doi: 10.1109/TPAMI.2017.2716350.

[419] Du, D., Qi, H., Wen, L., et al. *Geometric Hypergraph Learning for Visual Tracking.* 2016. url: http://arxiv.org/pdf/1603.05930v1.

[420] Yeo, D., Son, J., Han, B., et al. "Superpixel-Based Tracking-by-Segmentation Using Markov Chains". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2017, pp. 511–520. isbn: 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.62.

[421] Wang, J., Fei, C., Zhuang, L., et al. "Part-based multi-graph ranking for visual tracking". In: *2016 IEEE International Conference on Image Processing.* Piscataway, NJ: IEEE, 2016, pp. 1714–1718. isbn: 978-1-4673-9961-6. doi: 10.1109/ICIP.2016.7532651.

[422] Yao, R., Shi, Q., Shen, C., et al. "Part-Based Robust Tracking Using Online Latent Structured Learning". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.6 (2017), pp. 1235–1248. issn: 1051-8215. doi: 10.1109/TCSVT.2016.2527358.

[423] Zhang, B., Li, Z., Perina, A., et al. "Adaptive Local Movement Modeling for Robust Object Tracking". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.7 (2017), pp. 1515–1526. issn: 1051-8215. doi: 10.1109/TCSVT.2016.2540978.

[424] Ath, G. D. and Everson, R. *Part-Based Tracking by Sampling.* 2018. url: http://arxiv.org/pdf/1805.08511v1.

[425] Li, F., Jia, X., Xiang, C., et al. "Visual tracking with structured patch-based model". In: *Image and Vision Computing* 60 (2017), pp. 124–133. issn: 02628856. doi: 10.1016/j.imavis.2017.01.003.

[426] Zhang, S., Yao, H., Sun, X., et al. "Sparse coding based visual tracking: Review and experimental comparison". In: *Pattern Recognition* 46.7 (2013), pp. 1772–1788. issn: 00313203. doi: 10.1016/j.patcog.2012.10.006.

[427] Zhong, W., Lu, H., and Yang, M.-H. "Robust object tracking via sparsity-based collaborative model". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2012, pp. 1838–1845. isbn: 978-1-4673-1228-8. doi: 10.1109/CVPR.2012.6247882.

[428] Zhang, T., Liu, S., Ahuja, N., et al. "Robust Visual Tracking Via Consistent Low-Rank Sparse Learning". In: *International Journal of Computer Vision* 111.2 (2015), pp. 171–190. issn: 0920-5691. doi: 10.1007/s11263-014-0738-0.

[429] Zhang, T., Liu, S., Xu, C., et al. "Structural Sparse Tracking". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Piscataway, NJ: IEEE, 2015, pp. 150–158. isbn: 978-1-4673-6964-0. doi: 10.1109/CVPR.2015.7298610.

[430] Zhang, T., Ghanem, B., Liu, S., et al. “Robust Visual Tracking via Exclusive Context Modeling”. In: *IEEE transactions on cybernetics* 46.1 (2016), pp. 51–63. doi: 10.1109/TCYB.2015.2393307.

[431] Yi, Y., Cheng, Y., and Xu, C. “Visual tracking based on hierarchical framework and sparse representation”. In: *Multimedia Tools and Applications* 77.13 (2018), pp. 16267–16289. issn: 1380-7501. doi: 10.1007/s11042-017-5198-4.

[432] Guo, J., Xu, T., Shen, Z., et al. “Visual Tracking via Sparse Representation with Reliable Structure Constraint”. In: *IEEE Signal Processing Letters* (2017), p. 1. issn: 1070-9908. doi: 10.1109/LSP.2016.2645819.

[433] Javanmardi, M. and Qi, X. *Robust Structured Multi-task Multi-view Sparse Tracking.* 2018. url: http://arxiv.org/pdf/1806.01985v1.

[434] Zhu, G., Porikli, F., and Li, H. *Beyond Local Search: Tracking Objects Everywhere with Instance-Specific Proposals.* 2016. url: http://arxiv.org/pdf/1605.01839v1.

[435] Gao, C., Chen, F., Yu, J.-G., et al. “Robust Visual Tracking Using Exemplar-Based Detectors”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.2 (2017), pp. 300–312. issn: 1051-8215. doi: 10.1109/TCSVT.2015.2513700.

[436] Zhang, L., Varadarajan, J., Suganthan, P. N., et al. “Robust Visual Tracking Using Oblique Random Forests”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2017, pp. 5825–5834. isbn: 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.617.

[437] Henriques, J. F., Caseiro, R., Martins, P., et al. “Exploiting the Circulant Structure of Tracking-by-Detection with Kernels”. In: *Computer vision - ECCV 2012.* Ed. by Fitzgibbon, A., Lazebnik, S., Perona, P., et al. Vol. 7575. Lecture Notes in Computer Science. Berlin: Springer, 2012, pp. 702–715. isbn: 978-3-642-33764-2. doi: 10.1007/978-3-642-33765-9_50.

[438] Danelljan, M., Hager, G., Khan, F. S., et al. “Discriminative Scale Space Tracking”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.8 (2017), pp. 1561–1575. issn: 0162-8828. doi: 10.1109/TPAMI.2016.2609928.

[439] Tang, M. and Feng, J. “Multi-kernel Correlation Filter for Visual Tracking”. In: *2015 IEEE International Conference on Computer Vision.* Piscataway, NJ: IEEE, 2015, pp. 3038–3046. isbn: 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.348.

[440] Wibowo, S. A., Lee, H., Kim, E. K., et al. “Multi-scale color features based on correlation filter for visual tracking”. In: *International Conference on Signals and Systems.* Piscataway, NJ: IEEE, 2017, pp. 272–277. isbn: 978-1-5090-6748-0. doi: 10.1109/ICSIGSYS.2017.7967055.

[441] Danelljan, M., Häger, G., Shahbaz Khan, F., et al. “Accurate Scale Estimation for Robust Visual Tracking”. In: *Proceedings of the British Machine Vision Conference 2014.* Ed. by Valstar, M., French, A., and Pridmore, T. British Machine Vision Association, 2014, pp. 65.1–65.11. isbn: 1-901725-52-9. doi: 10.5244/C.28.65.

[442] Arfken, G. B. and Weber, H.-J. *Mathematical methods for physicists.* 6. ed., internat. ed., [5. Nachdr.] Amsterdam: Elsevier Acad. Press, 2008. isbn: 0-12-059876-0.

[443] Wong, M. W. *Discrete Fourier Analysis.* Vol. 5. Pseudo-Differential Operators, Theory and Applications. Basel: Springer Basel AG, 2011. isbn: 978-3-0348-0115-7. doi: 10.1007/978-3-0348-0116-4. url: http://dx.doi.org/10.1007/978-3-0348-0116-4.

[444] Duhamel, P., Piron, B., and Etcheto, J. M. “On computing the inverse DFT”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36.2 (1988), pp. 285–286. issn: 00963518. doi: 10.1109/29.1519.

[445] Qi, Y., Zhang, S., Qin, L., et al. “Hedged Deep Tracking”. In: *29th IEEE Conference on Computer Vision and Pattern Recognition.* Piscataway, NJ: IEEE, 2016, pp. 4303–4311. isbn: 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.466.

[446] Song, Y., Ma, C., Gong, L., et al. *CREST: Convolutional Residual Learning for Visual Tracking.* 2017. url: http://arxiv.org/pdf/1708.00225v1.

[447] Schölkopf, B. and Smola, A. J. *Learning with kernels: Support vector machines, regularization, optimization, and beyond.* Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2002. isbn: 9780262194754. url: http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=78092.

[448] Bolme, D. S., Draper, B. A., and Beveridge, J. R. “Average of Synthetic Exact Filters”. In: *CVPR 2009.* Los Alamitos, California: IEEE, 2009, pp. 2105–2112. isbn: 978-1-4244-3992-8. doi: 10.1109/CVPR.2009.5206701.

[449] Danelljan, M., Khan, F. S., Felsberg, M., et al. “Adaptive Color Attributes for Real-Time Visual Tracking”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2014, pp. 1090–1097. isbn: 978-1-4799-5118-5. doi: 10.1109/CVPR.2014.143.

[450] Zhang, K., Zhang, L., Liu, Q., et al. “Fast Visual Tracking via Dense Spatio-temporal Context Learning”. In: *Computer vision - ECCV 2014.* Ed. by Fleet, D. Vol. 8693. Lecture Notes in Computer Science. Cham: Springer, 2014, pp. 127–141. isbn: 978-3-319-10601-4. doi: 10.1007/978-3-319-10602-1_9.

[451] Li, Y. and Zhu, J. “A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration”. In: *Computer vision - ECCV 2014 Workshops.* Ed. by Agapito, L. Vol. 8926. Lecture Notes in Computer Science. Cham: Springer, 2015, pp. 254–265. isbn: 978-3-319-16180-8. doi: 10.1007/978-3-319-16181-5_18.

[452] Mueller, M., Smith, N., and Ghanem, B. “Context-Aware Correlation Filter Tracking”. In: *2017 IEEE Conference on Computer Vision and Pattern Recog-*

nition (CVPR). IEEE, 2017, pp. 1387–1395. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.152.

[453] ZHANG, M., XING, J., GAO, J., et al. "Joint Scale-Spatial Correlation Tracking with Adaptive Rotation Estimation". In: *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2015, pp. 595–603. ISBN: 978-1-4673-9711-7. DOI: 10.1109/ICCVW.2015.81.

[454] ZHANG, M., XING, J., GAO, J., et al. "Robust visual tracking using joint scale-spatial correlation filters". In: *2015 IEEE International Conference on Image Processing (ICIP)*. Piscataway, NJ: IEEE, 2015, pp. 1468–1472. ISBN: 978-1-4799-8339-1. DOI: 10.1109/ICIP.2015.7351044.

[455] HONG, Z., CHEN, Z., WANG, C., et al. "MUlti-Store Tracker (MUSTer): A cognitive psychology inspired approach to object tracking". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway, NJ: IEEE, 2015, pp. 749–758. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298675.

[456] MA, C., YANG, X., ZHANG, C., et al. "Long-term correlation tracking". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway, NJ: IEEE, 2015, pp. 5388–5396. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7299177.

[457] BERTINETTO, L., VALMADRE, J., GOLODETZ, S., et al. *Staple: Complementary Learners for Real-Time Tracking*. 2016. URL: http://arxiv.org/pdf/1512.01355v2.

[458] GALOOGAHI, H. K., FAGG, A., and LUCEY, S. *Learning Background-Aware Correlation Filters for Visual Tracking*. 2017. URL: http://arxiv.org/pdf/1703.04590v2.

[459] DANELLJAN, M., HAGER, G., KHAN, F. S., et al. "Learning Spatially Regularized Correlation Filters for Visual Tracking". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 4310–4318. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.490.

[460] LIU, T., WANG, G., and YANG, Q. "Real-time part-based visual tracking via adaptive correlation filters". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway, NJ: IEEE, 2015, pp. 4902–4912. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7299124.

[461] LIU, T., WANG, G., YANG, Q., et al. "Part-based Tracking via Discriminative Correlation Filters". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2017), p. 1. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2016.2637798.

[462] LI, Y., ZHU, J., and HOI, S. C. "Reliable Patch Trackers: Robust visual tracking by exploiting reliable patches". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Piscataway, NJ: IEEE, 2015, pp. 353–361. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298632.

[463] CHEN, W., ZHANG, K., and LIU, Q. "Robust visual tracking via patch based kernel correlation filters with adaptive multiple feature ensemble". In: *Neurocomputing* 214 (2016), pp. 607–617. ISSN: 09252312. DOI: 10.1016/j.neucom.2016.06.048.

[464] CHEN, K., TAO, W., and HAN, S. "Visual object tracking via enhanced structural correlation filter". In: *Information Sciences* 394-395 (2017), pp. 232–245. ISSN: 00200255. DOI: 10.1016/j.ins.2017.02.012.

[465] LIU, S., ZHANG, T., CAO, X., et al. "Structural Correlation Filter for Robust Visual Tracking". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 4312–4320. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.467.

[466] LI, C.-B., YANG, B., and LI, C.-H. "Deep Learning Based Visual Tracking: A Review". In: *2nd International Conference on Software, Multimedia and Communication Engineering (SMCE 2017)* (2017).

[467] FENG, X., MEI, W., and HU, D. "A Review of Visual Tracking with Deep Learning". In: *Proceedings of the 2016 2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE 2016)*. Paris, France: Atlantis Press, 2016. ISBN: 978-94-6252-271-8. DOI: 10.2991/aiie-16.2016.54.

[468] MA, C., HUANG, J.-B., YANG, X., et al. "Hierarchical Convolutional Features for Visual Tracking". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 3074–3082. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.352.

[469] CHI, Z., LI, H., LU, H., et al. "Dual Deep Network for Visual Tracking". In: *IEEE Transactions on Image Processing* 26.4 (2017), pp. 2005–2015. ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2669880. URL: http://arxiv.org/pdf/1612.06053v1.

[470] WANG, M., LIU, Y., and HUANG, Z. *Large Margin Object Tracking with Circulant Feature Maps*. 2017. URL: http://arxiv.org/pdf/1703.05020v2.

[471] ZHANG, T., XU, C., and YANG, M.-H. "Multi-task Correlation Particle Filter for Robust Object Tracking". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 4819–4827. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.512.

[472] WANG, N. and YEUNG, D.-Y. "Learning a Deep Compact Image Representation for Visual Tracking". In: *Advances in Neural Information Processing Systems 26*. Ed. by BURGES, C. J. C., BOTTOU, L., WELLING, M., et al. Curran Associates, Inc, 2013, pp. 809–817. URL: http://papers.nips.cc/paper/5192-learning-a-deep-compact-image-representation-for-visual-tracking.pdf.

[473] KAIHUA, Z., QINGSHAN, L., YI, W., et al. "Robust Visual Tracking via Convolutional Networks Without Training". In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 25.4 (2016), pp. 1779–1792. DOI: 10.1109/TIP.2016.2531283.

[474] GAO, J., ZHANG, T., YANG, X., et al. "Deep Relative Tracking". In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* (2017). DOI: 10.1109/TIP.2017.2656628.

[475] NAM, H., BAEK, M., and HAN, B. *Modeling and Propagating CNNs in a Tree Structure for Visual*

Tracking. 2016. URL: http://arxiv.org/pdf/1608.07242v1.

[476] Hong, S., You, T., Kwak, S., et al. *Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network.* 2015. URL: http://arxiv.org/pdf/1502.06796v1.

[477] Bertinetto, L., Valmadre, J., Henriques, J. F., et al. *Fully-Convolutional Siamese Networks for Object Tracking.* 2016. URL: http://arxiv.org/pdf/1606.09549v2.

[478] Guo, Q., Feng, W., Zhou, C., et al. "Learning Dynamic Siamese Network for Visual Object Tracking". In: *2017 IEEE International Conference on Computer Vision (ICCV).* IEEE, 2017, pp. 1781–1789. ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.196.

[479] Held, D., Thrun, S., and Savarese, S. *Learning to Track at 100 FPS with Deep Regression Networks.* 2016. URL: http://arxiv.org/pdf/1604.01802v2.

[480] Son, J., Baek, M., Cho, M., et al. "Multi-object Tracking with Quadruplet Convolutional Neural Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2017, pp. 3786–3795. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.403.

[481] Valmadre, J., Bertinetto, L., Henriques, J. F., et al. *End-to-end representation learning for Correlation Filter based tracking.* 2017. URL: http://arxiv.org/pdf/1704.06036v1.

[482] Vondrick, C., Shrivastava, A., Fathi, A., et al. *Tracking Emerges by Colorizing Videos.* 2018. URL: http://arxiv.org/pdf/1806.09594v2.

[483] Li, H., Li, Y., and Porikli, F. "Robust Online Visual Tracking with a Single Convolutional Neural Network". In: *Computer vision - ACCV 2014.* Ed. by Cremers, D., Reid, I., Saito, H., et al. Vol. 9007. Lecture notes in computer science Image processing, computer vision, pattern recognition, and graphics. Cham: Springer, 2015, pp. 194–209. ISBN: 978-3-319-16813-5. DOI: 10.1007/978-3-319-16814-2_13.

[484] Li, H., Li, Y., and Porikli, F. "DeepTrack: Learning Discriminative Feature Representations Online for Robust Visual Tracking". In: *IEEE Transactions on Image Processing* 25.4 (2016), pp. 1834–1848. ISSN: 1057-7149. DOI: 10.1109/TIP.2015.2510583. URL: http://arxiv.org/pdf/1503.00072v1.

[485] Sun, C., Wang, D., Lu, H., et al. *Learning Spatial-Aware Regressions for Visual Tracking.* 2018. URL: http://arxiv.org/pdf/1706.07457v2.

[486] Teng, Z., Xing, J., Wang, Q., et al. "Robust Object Tracking Based on Temporal and Spatial Deep Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV).* IEEE, 2017, pp. 1153–1162. ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.130.

[487] Wang, L., Ouyang, W., Wang, X., et al. "STCT: Sequentially Training Convolutional Networks for Visual Tracking". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2016, pp. 1373–1381. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.153.

[488] Gan, Q., Guo, Q., Zhang, Z., et al. *First Step toward Model-Free, Anonymous Object Tracking with Recurrent Neural Networks.* 2015. URL: http://arxiv.org/pdf/1511.06425v2.

[489] Milan, A., Rezatofighi, S. H., Dick, A., et al. *Online Multi-Target Tracking Using Recurrent Neural Networks.* 2016. URL: http://arxiv.org/pdf/1604.03635v2.

[490] Ning, G., Zhang, Z., Huang, C., et al. *Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking.* 2016. URL: http://arxiv.org/pdf/1607.05781v1.

[491] Choi, J., Kwon, J., and Lee, K. M. *Real-time visual tracking by deep reinforced decision making.* 2018. URL: http://arxiv.org/pdf/1702.06291v2.

[492] Huang, C., Lucey, S., and Ramanan, D. *Learning Policies for Adaptive Tracking with Deep Feature Cascades.* 2017. URL: http://arxiv.org/pdf/1708.02973v2.

[493] Sun, M. Z., Yen, C.-Y., and Lin, H.-S. *Visual Tracking using Deep Reinforcement Learning.* 2018.

[494] Yun, S., Choi, J., Yoo, Y., et al. "Action-Driven Visual Object Tracking With Deep Reinforcement Learning". In: *IEEE transactions on neural networks and learning systems* 29.6 (2018), pp. 2239–2252. DOI: 10.1109/TNNLS.2018.2801826.

[495] Zhang, D., Maei, H., Wang, X., et al. *Deep Reinforcement Learning for Visual Object Tracking in Videos.* 2017. URL: http://arxiv.org/pdf/1701.08936v2.

[496] Yi, S., Li, H., and Wang, X. "Pedestrian Behavior Understanding and Prediction with Deep Neural Networks". In: *Computer Vision – ECCV 2016.* Ed. by Leibe, B., Matas, J., Sebe, N., et al. Cham: Springer International Publishing, 2016, pp. 263–279. ISBN: 978-3-319-46448-0.

[497] Alahi, A., Goel, K., Ramanathan, V., et al. "Social LSTM: Human Trajectory Prediction in Crowded Spaces". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2016, pp. 961–971. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.110.

[498] Zhai, M., Roshtkhari, M. J., and Mori, G. *Deep Learning of Appearance Models for Online Object Tracking.* 2016. URL: http://arxiv.org/pdf/1607.02568v1.

[499] Wang, L., Ouyang, W., Wang, X., et al. "Visual Tracking with Fully Convolutional Networks". In: *2015 IEEE International Conference on Computer Vision (ICCV).* IEEE, 2015, pp. 3119–3127. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.357.

[500] Wang, N., Li, S., Gupta, A., et al. *Transferring Rich Feature Hierarchies for Robust Visual Tracking.* 2015. URL: http://arxiv.org/pdf/1501.04587v2.

[501] Jin, J., Dundar, A., Bates, J., et al. "Tracking with deep neural networks". In: *2013 47th Annual Conference on Information Sciences and Systems (CISS).* Piscataway, NJ: IEEE, 2013, pp. 1–5. ISBN: 978-1-4673-5239-0. DOI: 10.1109/CISS.2013.6552287.

[502] Nam, H. and Han, B. *Learning Multi-Domain Convolutional Neural Networks for Visual Tracking.* 2016. URL: http://arxiv.org/pdf/1510.07945v2.

[503] Ma, C., Huang, J.-B., Yang, X., et al. *Robust Visual Tracking via Hierarchical Convolutional Features.* 2018. URL: http://arxiv.org/pdf/1707.03816v2.

[504] Sutskever, I. *Training Recurrent Neural Networks.* Canadian theses = Thèses canadiennes. Ottawa: Library and Archives Canada = Bibliothèque et Archives Canada, 2014. ISBN: 0499220668.

[505] Sadeghian, A., Alahi, A., and Savarese, S. *Tracking The Untrackable: Learning To Track Multiple Cues with Long-Term Dependencies.* 2017. URL: http://arxiv.org/pdf/1701.01909v2.

[506] Fan, H. and Ling, H. "SANet: Structure-Aware Network for Visual Tracking". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).* IEEE, 2017, pp. 2217–2224. ISBN: 978-1-5386-0733-6. DOI: 10.1109/CVPRW.2017.275.

[507] Wang, L., Liu, T., Wang, B., et al. *Learning Hierarchical Features for Visual Object Tracking with Recursive Neural Networks.* 2018. URL: http://arxiv.org/pdf/1801.02021v1.

[508] Fang, K. *Track-RNN: Joint Detection and Tracking Using Recurrent Neural Networks.* 2016.

[509] Yang, T. and Chan, A. B. *Recurrent Filter Learning for Visual Tracking.* 2017. URL: http://arxiv.org/pdf/1708.03874v1.

[510] Fan, J., Xu, W., Wu, Y., et al. "Human tracking using convolutional neural networks". In: *IEEE transactions on neural networks* 21.10 (2010), pp. 1610–1623. DOI: 10.1109/TNN.2010.2066286.

[511] Vincent, P., Larochelle, H., Lajoie, I., et al. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *J. Mach. Learn. Res.* 11 (2010), pp. 3371–3408. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1756006.1953039.

[512] Hua, W., Mu, D., Guo, D., et al. "Visual tracking based on stacked Denoising Autoencoder network with genetic algorithm optimization". In: *Multimedia Tools and Applications* 77.4 (2018), pp. 4253–4269. ISSN: 1380-7501. DOI: 10.1007/s11042-017-4702-1.

[513] Zhou, X., Xie, L., Zhang, P., et al. "An ensemble of deep neural networks for object tracking". In: *2014 IEEE International Conference on Image Processing (ICIP).* IEEE, 2014, pp. 843–847. ISBN: 978-1-4799-5751-4. DOI: 10.1109/ICIP.2014.7025169.

[514] Chaudhuri, K., Freund, Y., and Hsu, D. *A parameter-free hedging algorithm.* 2010. URL: http://arxiv.org/pdf/0903.2851v2.

[515] Wang, N., Zhou, W., Tian, Q., et al. "Multi-Cue Correlation Filters for Robust Visual Tracking". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2018.

[516] Danelljan, M., Hager, G., Khan, F. S., et al. "Convolutional Features for Correlation Filter Based Visual Tracking". In: *2015 IEEE International Conference on Computer Vision Workshop (ICCVW).* IEEE, 2015, pp. 621–629. ISBN: 978-1-4673-9711-7. DOI: 10.1109/ICCVW.2015.84.

[517] Chatfield, K., Simonyan, K., Vedaldi, A., et al. *Return of the Devil in the Details: Delving Deep into Convolutional Nets.* 2014. URL: http://arxiv.org/pdf/1405.3531v4.

[518] Gladh, S., Danelljan, M., Khan, F. S., et al. *Deep Motion Features for Visual Tracking.* 2016. URL: http://arxiv.org/pdf/1612.06615v1.

[519] Malcolm, N. and Gibson, J. J. "The Perception of the Visual World". In: *The Philosophical Review* 60.4 (1951), p. 594. ISSN: 00318108. DOI: 10.2307/2181436.

[520] Danelljan, M., Robinson, A., Khan, F., et al. "Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking". In: (2016).

[521] Choi, J., Chang, H. J., Yun, S., et al. "Attentional Correlation Filter Network for Adaptive Visual Tracking". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2017, pp. 4828–4837. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.513.

[522] Cui, Z., Xiao, S., Feng, J., et al. "Recurrently Target-Attending Tracking". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2016, pp. 1449–1458. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.161.

[523] Simonyan, K., Vedaldi, A., and Zisserman, A. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.* 2014. URL: http://arxiv.org/pdf/1312.6034v2.

[524] Bromley, J., Guyon, I., LeCun, Y., et al. "Signature Verification Using a Siamese Time Delay Neural Network". In: *Proceedings of the 6th International Conference on Neural Information Processing Systems.* NIPS'93. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 1993, pp. 737–744.

[525] Taigman, Y., Yang, M., Ranzato, M., et al. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2014, pp. 1701–1708. ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.220.

[526] Zagoruyko, S. and Komodakis, N. *Learning to Compare Image Patches via Convolutional Neural Networks.* 2015. URL: http://arxiv.org/pdf/1504.03641v1.

[527] Jia, Y., Shelhamer, E., Donahue, J., et al. *Caffe: Convolutional Architecture for Fast Feature Embedding.* 2014. URL: http://arxiv.org/pdf/1408.5093v1.

[528] Han, B., Sim, J., and Adam, H. "BranchOut: Regularization for Online Ensemble Tracking with Convolutional Neural Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2017, pp. 521–530. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.63.

[529] Kristan, M., Leonardis, A., Matas, J., et al. "The Visual Object Tracking VOT2017 Challenge Results". In: *2017 IEEE International Conference on Computer Vision workshops.* Piscataway, NJ: IEEE, 2017, pp. 1949–1972. ISBN: 978-1-5386-1034-3. DOI: 10.1109/ICCVW.2017.230.

[530] Szepesvari, C. *Algorithms for reinforcement learning.* Vol. 9. Synthesis Lectures on Artificial Intelligence and Machine Learning. San Rafael, Calif.: Morgan & Claypool, 2010. isbn: 9781608454938.

[531] Konda, V. R. and Tsitsiklis, J. N. "OnActor-Critic Algorithms". In: *SIAM Journal on Control and Optimization* 42.4 (2003), pp. 1143–1166. issn: 0363-0129. doi: `10.1137/S0363012901385691`.

Appendix A

Object Detection (Supplementary Material)

The following section provides supplementary material for the object detection fundamentals in section 3.1. First, an overview of different base-architectures, which are used in modern convolutional object detectors, is given. Afterwards, frequently used object detection datasets and common performance metrics are presented. Finally, a thorough literature review on modern object detection algorithms is conducted.

A.1 CNN Base-Architectures

Typically, convolutional object detectors can be divided into a base network, which conducts the region proposal and feature extraction steps of the object detection pipeline and the detector network, which handles bounding box regression and object classification [28]. The base network is usually a pre-trained image classsification network, such as AlexNet [286], ZFNet [287], VGG-16 [288], GoogLeNet [289], Inception v2 [290], Inception v3 [291], ResNet-101 and ResNet-152 [177], Inception-ResNet v2 [176] and more recently MobileNets [292], DenseNet [293] and ResNeXt [294]. Depending on the detector, these base networks are used in slightly modified versions, ommiting at least the final classification layers and sometimes also intermediate layers or being fine-tuned on additional detection-specific datasets. Usually pretraining of these networks is conducted on large image classification datasets, such as ImageNet [295], which contains over 14 million annotated images of 1000 different categories, or Cifar-10 and Cifar-100 [296], each containing 60000 labeled images of 10 and 100 classes respectively. The choice of the base network is crucial for the detector accuracy and has a big impact on the computational complexity of the dectector algorithm as described in [119]. For example R-CNN uses the very large VGG-16 model with 138 million parameters as base network, which leads to a very long computation time for a single image, making this algorithm not applicable to real-time processing. On the other hand, such as YOLO [297] and YOLOv2 [298] use much smaller base networks, which lead to a significantly lower processing time, making those algorithms useful for real-time processing. In most cases, the same object detector can be used with different base networks. This allows for adoption of a detector to different usecases. For instance, YOLO has also been deployed with a variety of other base networks, such as VGG-16, AlexNet, ResNet and DenseNet [297, 299].

A.2 Detection Datasets

Frequently used publicly available datasets in object detection research as well as their according statistics are listed in table A.1. Provided are the total number of images containing bounding box annotations, the number of object categories (classes) and the total number of labeled objects. The statistics are combined over the contained training, validation and test datasets. The most popular and established datasets for object detection are Microsoft COCO (Common Object in Context) and Pascal VOC 2007 and 2012 (Visual Object Classes Chellenge). As most research papers in object detection use one of these three datasets, comparability between different studies is good. However, it has to be noted, that most of the datasets change every year to provide a state-of-the-art database for the accompanying yearly object recognition challenges. The Caltech 101 and Caltech 256 datasets are used more seldomly due to known issues [300, 301] and their relatively low number of training instances. On the other hand, datasets such as ImageNet ILSVR and the Open Images Dataset are used less frequently due to their large size, which makes them difficult to use, if only a low-performant desktop workstation is availabe for training.

Tab. A.1: Overview and statistics of frequently used datasets for object detection.

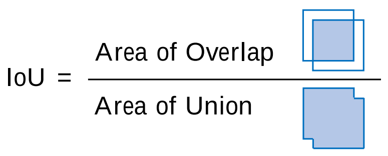
Dataset	Year	Images	Classes	Labeled Instances	References
Microsoft COCO	2015	328124	80	~1500000	[302]
Pascal VOC	2007	14974	20	37248	[303]
Pascal VOC	2012	11540	20	31561	[304]
ImageNet ILSVRC	2014	516840	200	1068618	[305]
Open Images Dataset V4	2018	1910098	600	15440132	[306]
Caltech Pedestrian Detection	2012	~250000	1	~ 350000	[307], [308]
KITTI Vision Benchmark	2012	14999	2	80256	[169]
Caltech 101	2003	9146	101	n.A.	[309]
Caltech 256	2007	30607	256	n.A.	[310]

Most object detectors are trained not only on the raw samples of these datasets, but on heavily augmented versions of the datasets. Data augmentation means, additional training samples are generated by techniques, such as random rescaling, cropping, translating, rotating and flipping of the original samples. Often image parameters, such as brightness, contrastm, saturation and color are also adjusted to generate even more samples. Data augmentation is a frequently applied regularization strategy, which helps to prevent overfitting of neural networks to the training dataset during training by showing the network a larger amount of high-variance training data. [11, 297, 311, 312]

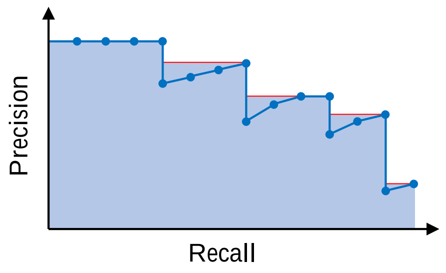
A.3 Performance Metrics

To be able to compare different object detection algorithms with each other, common performance metrics have to be defined. Here, the *mean average precision* (mAP) and the *intersection over union* (IoU) are established performance metrics in object detection research [142, 313, 314]. As depicted in fig. A.1a, the IoU is a measure that describes, how well a predicted bounding box aligns with the ground-truth bounding box of the dataset. It is the ratio between the area of overlap and the area of union of both boxes. In case, the predicted box lies exactly on the ground truth box, the IoU has a value of 1 and in case of no overlap it has a value of 0.

The exact definition of the mAP varies slightly between the object detection challenges. Hence, the following explanation applies only to Pascal VOC and has to be modified for other challenges, such as COCO. Refer to the references above to get the according definitions of the mAP. For Pascal VOC, the mAP is the mean of the average precision (AP) over all object classes. The AP can be calculated as area under the interpolated precision-recall curve of the classifier as shown in fig. A.1b [315]. To compute this curve, true positives, false positives and false negatives for all bounding box matches have to be determined separately for each class, which is done by sorting matches in decreasing order of their class label confidences. Here, a match is counted as a true positive, when the predicted bounding box and the ground-truth bounding box have the same class label and their IoU is greater or equal than 0.5. Accordingly a false negative is a labeled object in the dataset, that is not found by the detector, and a false positive an object, that has been found by the detector, but is not labeled as ground-truth object in the dataset. After their computation, the precision-recall curves are interpolated, which means, a monotonic decreasing curve is computed (red lines) by setting the precision value p for recall r to the maximum precision $p_{\max}$ obtained for any recall value $r' > r$. The AP for each classs is then found by computing the area under the according precision-recall curve. The mean of all AP values finally yields the mAP, which lies in a range from 0 % to 100 %. [316]



(a) Definition of IoU.



(b) Definition of AP.

Fig. A.1: Definitions of Intersection over Union (IoU) and average precision (AP) performance metrics for quantitative analysis of object detection algorithms. [316]

A.4 Convolutional Object Detectors

After clarification of the basics, the most important convolutional object detectors are briefly introduced in this section. The list is ordered chronologically after the initial submission date of the proposing paper as can also be seen from the timeline in fig. A.2. For further details, refer to the mentioned references or the overviewing articles in [317], [318], [319], [320], [321] and [322]. A comparison of different architectures by means of accuracy and performance is given in [119], [323]. In the past years of research in this field, two groups of convolutional object detectors have evolved. The first group are region based object detectors, which utilize a region proposal algorithm, such as a sliding window, clustering or a region proposal network, to generate regions of interests, which are then fed into the further detection pipeline. Detectors, such as R-CNN, OverFeat, SPP-net, Fast R-CNN, Faster R-CNN and R-FCN are among this group. The other group are *one-shot detectors*, such as all of the YOLO detectors, SSD and RetinaNet. Instead of extracting and processing individual regions of interests, they operate on the entire input image at once and apply an internal discretization of the image to perform object localization. Typically, one-shot detectors are much faster than the region-based detectors, while having roughly the same accuracy.

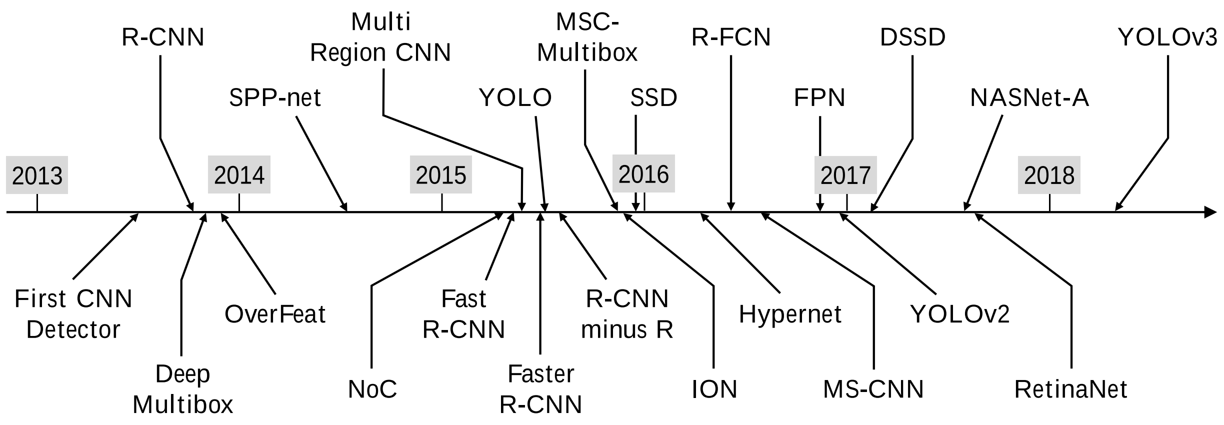


Fig. A.2: Timeline of the presented convolutional object detection algorithms. Reference is the submission date of the first version of the propsing papers.

R-CNN (2013) One of the first algorithms, that made use of convolutional networks was R-CNN. R-CNN uses the already mentioned selective search algorithm from [20] to generate around 2000 class-independet region proposals for bounding boxes of objects in each image. The image patches in these regions are mean-subtracted and resized to a fixed dimension of 227×227 pixels and then sequentially fed into an AlexNet [286] convolutional neural network for feature extraction. Features for all regions are then fed into a set of class-specific SVMs, which classify the object in each region. Moreover, the CNN features are used in an additional bounding box regression to reduce localization erros. This regression is not performed by a neural network, but instead formulated directly as ridge regression. R-CNN also implements non-maximum supression to prevent overlapping bounding boxes for the same object. Here, boxes with lower confidence scores for the class prediction are removed. In 2013 R-CNN had a state-of-the-art accuracy with an mAP of 58.5 % on Pascal VOC 2007 and 53.3 % on Pascal VOC 2012, however the algorithm is very slow. [142]

OverFeat (2013) Another early dependant of convolutional object detectors is OverFeat, which combines the tasks of image classification, object localization and object detection in a single convolutional neural network and shows the superior performance of this approach compared to earlier object detection methods. The model is trained in three steps, beginning with the classification CNN, which consists of 5 convolutional layers and 3 fully connected layers. After training of this network on ImageNet 2012, the convolutional layers are used to extract features of resized input images by means of a sliding window approach on a generated image pyramid. These features are fed into a fully connected 2-layer regression network, that realizes bounding box regression to localize objects. By utilizing negative training examples (image background), the feature extraction layers are fine-tuned for the detection task. The bounding box regression network is simply reused for detection. Instead of performing NMS on found bounding boxes, boxes with the highest class confidences are merged. OverFeat showed a state-of-0the-art performance in 2013 and scored ranks 4, 1 and 1 for image classification, localization and detection in the 2013 ImageNet Large Scale Visual Reecognition Competition. However, training of the three independent neural networks is complicated and computational performance of the algorithm is also low. [314, 324]

SPP-net (2014) This detector is very similar to R-CNN, however tries to speed up the original algorithm by extracting convolutional features in a parallel manner, rather than sequentially. Instead of feeding each proposed region into the CNN separately, the entire image is fed into the CNN to generate a feature map. Only then, regions are extracted from the feature map and transformed into a fixed sized representation by a proposed spatial pooling layer. Classification of the objects is conducted by means of SVMs again, which means, SPP-net still requires sequential training of multiple sub-modules as it was the case with R-CNN, which makes the process complicated and leads to sub-optimal accuracy. However, due to the parallel feature extraction approach, SPP-net is multiple times faster than R-CNN. [325]

Fast R-CNN (2015) This detector is an improvement of R-CNN and SPP-net and provides a faster execution time and higher accuracy. Region proposal is still conducted by means of the selective search algorithm, however the network architecture utilized the same method as in SPP-net to speed up feature extraction. Instead of passing each proposed region into the feature extractor, the entire image is fed through and feature vectors are extracted from the resulting feature map according to the proposed regions. Every feature vector is then pooled with a spatial pooling layer and fed into fully connected layers, which do bounding box regression and, with the help of a softmax activation function, object classification. A multi-task loss function is used to train this network. Since feature extraction, bounding box regression and object classification are integrated in a single network, training can be done end-to-end, simplyfing the procedure. Fast R-CNN achieved state-of-the-art accuracy in 2014, with an mAP of 70.0 %

and 68.4 % on Pascal VOC 2007 and 2012 respectively, and significantly sped up inference on images. [317, 319, 326]

Faster R-CNN (2015) This detector is a further improvement of Fast R-CNN. It reduces inference time by replacing the computationally expensive selective search algorithm with a convolutional region proposal network (RPN) for generating bounding box proposals. This is done by sliding a small 3×3 CNN with a subsequent fully connected sibling-layer over the feature map, extracted from the input image, and generating 9 region proposals for each spatial position in this feature map. Each region proposal contains 4 box dimensions (width, height, horizontal position and vertical position), which are generated by the first fully connected layer, and 2 *objectness-scores*, which are generated by the second fully connected layer and encode probabilities for the box containing an object or not. The box coordinates are defined relative to 9 different *anchor-boxes* with different sizes and aspect ratios. The so generated region proposals are used similarly to the region proposals of the selective search algorithm in the Fast R-CNN algorithm. Object classsification and bounding box regression are identical to the approach in Fast R-CNN. A further improvement introduced in Faster R-CNN is weight-sharing between the RPN and the feature extraction network, which enables usage of a single convolutional neural network for region proposal, feature extraction, object classification and bounding box regression. Hence, faster R-CNN was the first object detection approach based on a single unified neural network. This lead to a higher mAP of 78.8 % and 75.9 % on Pascal VOC 2007 and 2012 respectively, and much faster inference compared to earlier methods. Moreover, Faster R-CNN allows for true end-to-end training, simplifying the training and testing procedure. [9]

YOLO (2015) Aim of the authors of this very popular object detector was to drastically reduce inference time to make their detector applicable to real-time tasks, such as video processing. They achieved this by refining the detection pipeline introduced in fig. 3.2. Instead of modelling the detection pipeline in form of individual components as done in the R-CNN variants, YOLO models the detection process as a single regression task, with the entire image as input and bounding box coordinates, box confidence and class probabilites as outputs. YOLO is a 24-layer, GoogLeNet-inspired [289], convolutional architecture with two additional 4096-dimensional fully connected layers for prediction of outputs. The final output has shape $S \times S \times (B * 5 + C)$, since the input image, which is resized to 448×448 pixels, is modelled as an $S \times S$ grid of equally sized cells and for each grid cell B bounding boxes with five parameters each (width, height, vertcial position, horizontal position, confidence score) as well as C class probabilities are predicted. The convolutional layers are pretrained on ImageNet 2012 images of size 224×224 pixels and the entire network, including the fully connected layers, is fine-tuned for detection on Pascal VOC 2007 and 2012 afterwards. Here, the grid size is set to $S = 7$ and two bounding boxes are generated for each cell ($B = 2$), which results in an output of size of $7 \times 7 \times 30$, since there are $C = 20$ classes in Pascal VOC. Hence, YOLO generates only 98 bounding boxes per image. For training a multi-part sum-squared error loss function is used. It penalizes wrong box positions and dimensions as well as wrong class labels in case the according box contains an object, which is indicated by a high box confidence score. As in the earlier object detection algorithms, non-maximum-supression is used to remove duplicate detections of the same object. Due to its one-shot approach for bounding box detection and object classsification, YOLO achieved high processing frame rates and was the first object detector that allowed for real-time object detection. Its mAP was 63.4 % on Pascal VOC 2007 and 57.9 % on Pascal VOC 2012. This is lower than the accuracy of Fast R-CNN and Faster R-CNN, however YOLO is 90 times faster than R-CNN (45 FPS versus 0.5 FPS) and 2.5 times faster than Faster R-CNN (45 FPS versus 18 FPS). Due to the fact, that YOLO processes the entire image at once, it sees objects in context and thus produces less false positives on the image background than the earlier detectors. One issue of YOLO are comparably high localization errors. Hence, YOLO has problems to detect small objects, that lie close to each other. [297, 319, 321]

Single Shot Detector (SSD) (2015) This is another real-time detector, that follows the approach of YOLO utilizing a single convolutional neural network to implement the entire detection pipeline in a unified model and hence allows for very fast predictions and an uncomplicated end-to-end training. SSD uses a truncated and modified version of VGG-16 [288] as a base network on which six additional convolutional layers with decreasing size are stacked. One major difference of SSD compared to YOLO is the simultaneous usage of feature maps of different scales, which are generated by the additional convolutional layers. Similar to Faster R-CNN, SSD uses multiple anchor boxes of different sizes and aspect ratios per feature map cell as a starting point for the bounding box regression. The anchor boxes, which are called *priors* in the SSD paper, are manually defined to match the distribution of ground-truth bounding box dimensions and positions in the training dataset. Output of SSD is a set of bounding box parameters and a vector of class probabilities for each of the 8872 generated bounding boxes per image. The base network is pretrained on ImageNet ILSVRC 2015. The entire detector network is then fine-tuned on either Pascal VOC 2007, 2012 or the COCO 2015 dataset. Here, the training loss is defined as weighted sum of a smooth L1-norm as localization loss, that penalizes misaligned boxes, and a softmax confidence loss, that penalizes wrong class predictions. To compute the losses between the prediction and the ground-truth boxes, all predicted boxes, which have an IoU higher or equal to 0.5 with the ground-truth box are matched. Predictions, which do not match the ground-truth boxes are used as negative training examples in a process called hard-negative-mining. Hereby, the ratio of positive to negative training examples is kept constant at around 1/3. SSD also uses non-maximum-supression to remove duplicate bounding boxes for the same object. The standard SSD network requires input images to have a fixed size of 300×300 pixels and achieves an mAP of 79.6 % on Pascal VOC 2012 and 77.5 % on Pascal VOC 2012. Another SSD, that operates on larger, 512×512 pixel sized, images was developed by adding an additional convolutional layer to the network input. This SSD512 network achieved a higher accuracy of 81.6 % on Pascal VOC 2012 and 80.0 % on Pascal VOC 2012, however at a lower processing frame rate of 22 FPS compared to 59 FPS of SSD300. [11, 319, 322]

R-FCN (2016) This detector follows the algorithmic approach of the R-CNN family using a region proposal network (RPN) to generate possible bounding box locations. Similar to Faster R-CNN a single convolutional neural network implements all steps of the object detection pipeline. To speed up computation in comparison to Faster R-CNN, R-FCN follows a different approach to generate bounding boxes and classify objects. Instead of sliding a small convolutional filter over the image and recomputing the convolutions every time, R-FCN takes the entire image as input and generates a set of $S \times S$ location sensitive score maps for each of the C object classes, yielding $S^2(C+1)$ different score maps (+1 because the background is counted as an additional class). Each score map corresponds to an object of a certain class being located a certain discrete image position. Hence, if there is an object of class *dog* in the upper-left corner of the input image, the score map responsible for the upper-left corner and the dog-class will show a high activation, while all other score maps have a low activation. By averaging over all S^2 score maps for a certain class, the detector can determine, whether the proposed region of interest overlaps well with the object of that class. The resulting $(C+1)$-dimensional vector can also be used to classify the object by computing softmax activation. R-FCN performs the same bounding box regression as utilized in Fast R-CNN to reduce localization errors. The convolutional network is a truncated ResNet-101 [177] which is pretrained on ImageNet 2015. Similar to Faster R-CNN, the RPN shares its weights with the feature map extractor and hence requires no additional computations. R-FCN also proposes 300 regions of interest per image. The entire algorithm is fine-tuned in an end-to-end fashion on Pascal VOC 2007, Pascal VOC 2012 and MS COCO and achieves an mAP of 83.6 % on the test sets of Pascal VOC 2007 and 82.0 % on Pascal VOC 2012, while being 2.5 to 20 times faster than R-CNN. [178, 319]

YOLOv2 and YOLO9000 (2016) YOLOv2 is a drastic improval of the YOLO object detector, addressing some of the issues of the original detector, such as high localization errors and low recall, while maintaining its high processing speed. YOLOv2 uses a redesigned ResNet-like base network, called Darknet-19, which contains 19 convolutional and 5 maxpooling layers and is pretrained on the ImageNet 2015 dataset. It contains only 5.58 million parameters (YOLO: 8.52 million) and hence runs significantly faster, but still provides a better classification accuracy on ImageNet. Apart from this, YOLOv2 uses batch normalization layers instead of dropout for regularization and is trained on images of various sizes, ranging from 320×320 to 608×608 pixels in steps of 32 pixels. This allows to use input images of different dimensions and hence enables an easy tradeoff between speed and accuracy as larger inputs images decrease processing speed, but increase accuracy. Another improvement used by YOLOv2 are anchor-boxes, as introduced in Faster R-CNN. Instead of directly regressing box coordinates from random intial values, 5 anchor boxes are defined per grid cell of the discretized input image. Dimensions of these anchor boxes are found by k-means clustering of the dimensions of all ground-truth boxes in the training dataset. Matching of anchor-boxes to predicted boxes is done by thresholding the IoU at a value of 0.5, as done in the SSD detector. The final layer of Darknet-19 outputs 4 box coordinates, 1 objectness score for the box and a vector of class probabilities (with 20 values in case of Pascal VOC). In YOLOv2 box coordinates are parameterized with respect to the dimensions of the matched anchor box and the position of the according grid cell, which stabilizes the trainign procedure and improves convergence speed. YOLOv2 also makes use of the finding of the SSD detector and uses feature maps of two different scalings as basis for the bounding box regression and object classification. This is done by passing through the 26×26 feature map of an earlier layer and concatenating this with the 13×13 feature map of the final layer. Training of YOLOv2 is done on Pascal VOC 2007 and 2012. The achieved performance is with an mAP of 78.6 % on Pascal VOC 2007 and 73.4 % on Pascal VOC 2012 very similar to this of Faster R-CNN and SSD512, however YOLOv2 is 2 to 10 times faster. The YOLOv2 paper also introduces another object detector, YOLO9000, which is based on the YOLOv2 architecture and can classify objects of 9000 different classes. This algorithm was trained by merging the 80 object classes of the MS COCO object detection dataset with the 1000 classes of the ImageNet image classification dataset. YOLO9000 is still real-time capable and achievs an mAP of 19.7 % on the ImageNet detection set. [298, 319, 321]

RetinaNet (2017) This is a one-shot object detector with relatively simple architecture, that utilizes a novel focal loss function to address the problem of class imbalance between a light and hard misclassified samples, and thus increase detection accuracy while maintaining the high processing speed of one-shot detectors. Focal loss is a modification of the standard cross-entropy loss, that reduces the incurred loss of well-classified samples, which otherwise overwhelms the influence of rare hard misclassified samples. The authors prove focal loss to be useful by building a simple convolutional object detector called RetinaNet, which utilizes a ResNet-101 base network that is augmented by a Feature Pyramid Network (FPN) [327] to generate a rich, multi-scale feature pyramid of a single input image. Each pyramid layer is connected to a convolutional sub-network for classification and a convolutional sub-network for bounding-box-regression. RetinaNet makes use of 9 anchor boxes with different sizes and aspect ratios as references for the bounding-box-regression. Anchors and ground-truth boxes are matched, if their IoU is equal or larger than 0.5. The classification sub-network predicts the objectness score and class probabilities for each bounding box. The base network is pretrained on the ImageNet 2015 dataset and fine tuning of the entire detector is performend on the MS COCO dataset. Here, the focal loss is used. The best RetinaNet model, in which the ResNet-101 base network is replaced by a ResNeXt-101 network, surpasses region proposal detectors, such as Faster R-CNN, by achieving an mAP of 40.8 % on the MS COCO dataset at a processing speed of 5 FPS. [10, 328]

YOLOv3 (2018) To bring accuracy of the YOLOv2 detector on a par with state-of-the-art models in 2018, such as RetinaNet, while still maintining YOLOs unchallegend processing speed, many improvements of the architecture were presented, yielding YOLOv3. The first improvement relates to the class prediction. Instead of predicting mutually exclusive classes as in YOLOv2, YOLOv3 can assign multiple class labels per bounding box, which increases accuracy in more complex datasets, such as Open Images Dataset, as this contains mutually non-exclusive labels, for example can an object be labeled as *human* and *child* simultaneously. Technically this was achieved by replacing the softmax classifier with multiple independent logistic classifiers and changing the mean-squared errors loss function to a binary cross-entropy loss. Another feature in YOLOv3 is the usage of pyramid features, which means, YOLOv3 predicts bounding box coordinates, the box confidence level and all classs probabilities at three different scales. This is achieved by concatenating the feature map of the final convolutional layer with two upsampled feature maps of earlier layers. These earlier layers lie two and four layers previous to the final layer. If the image is discretized to a grid of $S \times S$ cells, the final output tensor has a shape of $S \times S \times [3 * (4 + 1 + C)]$, where 4 indicates the dimension and location of the bounding box, 1 is the box confidence score, and $C = 80$ are the object classes in the MS COCO dataset. As in the earlier version, YOLOv3 still uses predefined anchor boxes as starting point for the bounding box regression. Now, 9 different anchor boxes, which were found by k-means clustering of the ground-truth boxes in the training dataset are used for each image location. Another change introduced in YOLOv3 was the Darknet-53 feature extractor, a 53-layer CNN with ResNet-like architecture. This network provides the same classification accuracy as ResNet-152, but halves the inference time. The skip-layers of Darknet-53 speed up and stabilize the training process. Similar to YOLOv2, YOLOv3 can operate on images of different sizes. With input images of size 608×608 pixels YOLOv3 achieves an mAP of $33.0\,\%$ on the COCO dataset (official COCO mAP metric). This is below the accuracy of RetinaNet, but one has to consider, that YOLOv3 is still 3.8 times faster than RetinaNet. [8, 321, 323]

Others Apart from the above presented object detectors, there exist a large number of different detectors and variants of the presented algorithms. However, these are cited and compared less often, hence they are mentioned more briefly in the following paragraphs.

The first work, using deep neural networks for object detection was conducted by Szegedy et al. [329]. Here, the idea of formulating localization of the bounding box as a regression problem was introduced. A sliding window extracted subregions of the image, which were fed into a CNN producing the bounding box. This process was iteratively repeated to refine the bounding box dimensions and location. As the presented detector could only localize one class at a time, it had to be separately applied to every class in the dataset.

Deep Multibox, developed by Erhan et al. [330], built upon those ideas by presenting a class-unaware detector based on a CNN. Rather than detecting bounding boxes for only a specific class, it generated bounding boxes for all objects in an image regardless of their class. Classification of the object in each box was then performed by means of a subsequent classification CNN. Deep Multibox also introduced the box confidence score, measuring the probability of a box containing an object or not.

This work of Ren et al. [331] built upon the SPP-net architecture and studied the importance of the classsifier performing the object classification based on the feature maps extracted from a proposed region of interest. They called the family of classifiers Networks on Convolutional feature maps (NoC) and conducted a series of experiments, in which they replaced the SVM classifiers of the original SPP-net architecture with different types of NoCs, such as different multi-layer perceptrons (MLPs), CNNs and CNN-pairs, which are merged by means of a maxout-layer. Their findings could be seen as an improvement of the SPP-net object detector and contributed to the general understanding of the classifier in the object detection pipeline.

Gidaris et al. [332] presented the Multi Region CNN algorithm, which was an object detector based on the principle of R-CNN. The work aimed at improving detection accuracy by generating richer feature maps. This was achieved by extracting not only one feature map for a proposed bounding box region, but also focus on 9 overlapping subregions of each bounding box, such as left half, right half, top, bottom, etc. and extract features for each of these subregions. For this, they built a unified CNN architecture with several parallel branches for each of the subregions, and surpassed even Fast R-CNN with an mAP of 78.2 % and 73.9 % on Pascal VOC 2007 and 2012, respectively.

Lenc et al. [333] presented R-CNN minus R, a streamlined version of R-CNN, that omitted the computationally expensive and difficult to train region proposal step and instead extracted features from 3000 primitive, constant regions and fed them into a CNN architecture for detection. It followed the SPP-net approach of generating feature maps of various different scales, while having been faster. However, its mAP on Pascal VOC 2007 was below the mAP of SPP-net, having shown, that region proposal or otherwise found bounding box priors are neccessary for a high detection accuracy.

A follow up work on the Deep Multibox detector was the Inception-based MSC-Multibox detector by Szegedy et al. [334]. It aimed at generating less region proposals with higher quality to reduce processing time in the later stages of the detector. Similar to Deep Multibox these region proposals were class-agnostic, which meant, boxes were generated for all objects in the image regardless of their class. Hence, their base algorithm was a region proposal network, that had to be paired with a classifier to be applicable to object detection tasks. MSC-Multibox allowed for a flexible number of region proposals per image and thus provided an easy to tweak trade-off between accuracy and processing time.

Bell et. al [335] developed the Inside-Outside Net (ION) for object detection. Aiming at higher detection accuracy, their architecture utilized contextual information of the object both within the objects region of interest as well as outside of it. They achieved this by having used two subsequent spatial recurrent neural networks, that passed information both vertically and horizontally accross the image and thus provided contextual information. Additionally, ION extracted feature maps of different scales to allow for detection of differently scaled objects. In Pascal VOC 2007 and 2012 ION achieved a mAP of 79.2 % and 76.4 %, repsectively.

Kong et al. [336] proposed HyperNet, a VGG-16-based CNN for joint region proposal and object detection. Similar to MSC-Multibox it aimed at reducing the number of region proposals to speedup the object detection process. Instead of proposing 2000 bounding boxes as R-CNN did, HyperNet proposed only 100 bounding boxes, which led to significantly faster processing speed, while having maintained a high mAP of 76.3 % on Pascal VOC 2007 and 71.4 % on Pascal VOC 2012. HyperNet used a shared feature map for both region proposal and object detection. This feature map was created by concatenating appropiately sampled feature maps of different layers and thus contained information about objects of various sizes.

MS-CNN was another region proposal-based object detector, introduced by Cai et al. [337]. It contained two sub-networks, a fully convolutional proposal network and a detection network, which shared some layers to reduce computations. The proposal network had a main-trunk from which several output layers branched off. The outputs of these branches were proposed object regions at different scales, which were then fed into the object detection network that performed bounding box regression and object classification. Training of MS-CNN was done end-to-end by means of a multi-task loss, that computed the proposal loss over all branches as well as the loss for bounding box regression and object classification. MS-CNN generated around 100 proposals per image and outperformed Faster R-CNN by means of detection accuracy, while having been 5 times faster on the KITTI Vision Benchmark.

Lin et al. [327] presented the Feature Pyramid Network (FPN), which was not directly an object detector, but a CNN feature extractor for compute- and memory-efficient generation of feature pyramids. Such feature pyramids were sets of feature maps of various scales and were already utilized by several of the previously introduced object detectors, such as SSD, SPP-net, ION and HyperNet. However, their implementations were less efficient than the FPN. The authors demonstrated their approach by extending Fast R-CNN and Faster R-CNN, which originally only utilized features of a single scale, with an FPN, having yielded significant improvements in detection accuracy on the MS COCO dataset.

Fu et al. [338] developed the Deconvolutional Single Shot Detector (DSSD), which was based on the SSD detector, however introduced significant changes to the original architecture in order to introduce additional context information into the object detection process. The core idea of the DSSD was to extend the original fully convolutional SSD architecture with several deconvolutional layers and merge feature maps from earlier convolutional layers with maps generated by the previos deconvolutional layer. This resulted in an hourglass-shaped neural network that had links between corresponding convolutional and deconvolutional layers. A set of identical convolutional prediction modules branched off each deconvolutional layer and conducted bounding box regression as well as object classification. A further change introduced in DSSD architecture was the usage of ResNet-101 instead of the original VGG-16 base network, because ResNet-101 achieved more accurate results. DSSD achieved an mAP of 81.5 % on Pascal VOC 2007 and 80.0 % on Pascal VOC 2012, however at the cost of an increased inference time due to the additional deconvolutional layers. As this detector took object context into account, it had a signifcant performance on small objects, which could be detected only due to their context, such as a tie on a person behing a lectern.

NASNet-A was an object detection base network, that resulted from the work of Zoph et al. [175] on automatic neural architecture search for image classification and object detection tasks. Instead of having manually engineered a base network, such as those presented in appendix A.1, their search system automatically found the NASNet-A architecture after a 4 day long search on a worker pool of 500 Nvidia P100 GPUs. They used NASNet-A as a base network for Faster R-CNN and achieved an mAP of 43.1 % on the MS COCO dataset.

Despite the large amount of presented object detection algorithms, there exist many more algorithms. These can easily be found in the leaderboards of the object detection challenges, such as the MS COCO detection challenge or the ImageNet ILSVRC.

Appendix B

Object Tracking (Supplementary Material)

The following section provides supplementary material for the object tracking fundamentals in section 3.2. First, challenges of object tracking are introduced. Afterwards, commonly used feature representation of the tracking targets are presented. Finally, an overview of the recent literature is given. Here, three main categories of object trackers are identified. Classical trackers, correlation filter based trackers and deep learning based trackers. For each category the most popular and most recent tracking algorithms are presented. Despite covering many tracking algorithms, this list is not exhaustive and for further reference the results of annual object tracking challenges, such as VOT [146] or MOT [7, 339], should be examined.

B.1 Challenges of Tracking

Tracking is accompanied by a number of challenges. The most important one is drift of the tracker over time. This drift has various reasons, such as occlusion of the tracked object, changes of object pose, scale and illumination, fast object movement and motion blur as well as camera movement relative to the tracked object [340–343]. Further challenges for the tracker are merging and splitting targets [34, 344, 345], targets entering or exiting the frame [346] and low resolution targets [347–349]. Especially in the context of human tracking more problems arise. For example in crowded scenes there are many similar looking objects close to each other, which can lead to jumping prediction results [350, 351]. Moreover, a person could drastically change appearance by taking off his or her coat, confusing the tracker [352]. And finally, if there are multiple cameras involved, target handover between different camera frames has to be handled [353–356].

B.2 Features for Tracking

The feature extraction module in the tracking pipeline (see section 3.2) deserves a closer look, as the ablation study in [37] showed that this module has the largest impact on tracking performance. Classical methods of feature extraction are covered exhaustively by Li et. al in a 2013 survey [340]. Such methods include templates, which are raw pixel-patches of the target region [357, 358], normalized color histograms [359–361] and multi-part color histograms [362], which combine color histograms of multiple overlapping target regions. Additionally, image gradients can be used to model target appearance [363–366], whereby sometimes a histogram of oriented gradients (HOG) is used to reduce the amount of data [367]. Instead of using histograms for representing color or gradient distributions, some methods utilize Gaussians models [368] or mixtures of Gaussians [369]. Other classically used features are the optical flow representation [370], covariance representations [371, 372], superpixel representation [373] and wavelet filters [374] as well as local features, such as SIFT- [375], MSER- [376], SURF- [377] and Harris corner features [378]. The purpose of all these methods is to reduce the tracking target to a representation, which is more robust against appearance changes of the target.

B.3 Classical Tracking Algorithms

This section briefly covers classical object tracking algorithms, which embraces all trackers, which do not utilize a correlation filter or any deep neural network, as these two categories will be covered later. For a taxonomy of tracking algorithms see fig. B.1.

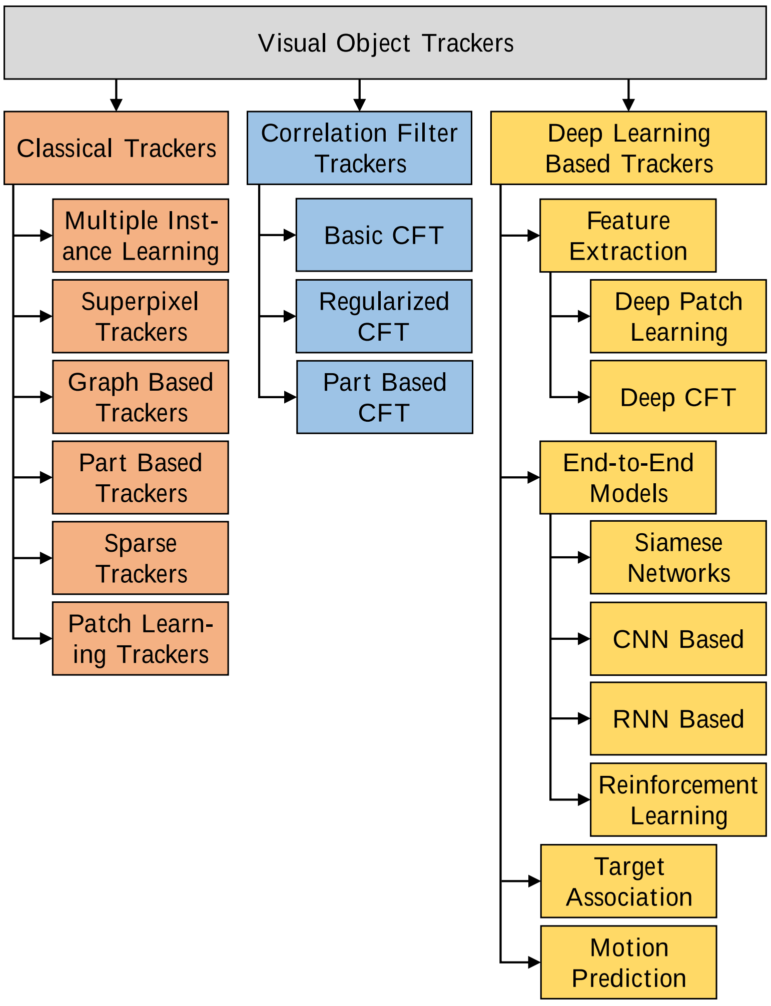


Fig. B.1: Categorization of object tracking algorithms. Classical object tracking approaches will be covered in this section, correlation filter based trackers in appendix B.4 and deep learning based trackers in appendix B.5. [38, 39]

There exist several excellent surveys covering classical tracking algorithms. Yilmaz et al. [379] provided a comprehensive review of object tracking algorithms in 2006, categorizing the existing trackers into *point trackers*, *kernel trackers* and *silhoutte trackers*. Point trackers receive a set of keypoints of the tracked object in every frame and they assign corresponding points across frames based on the target motion in previous frames, hence they comprise only a motion model, but no apearance model of the target. Examples are particle filters [380], the Kalman filter [381], the Joint Probability Data Association Filter (JPDAF) [382] and the Multiple Hypothesis Tracker (MHT) [44]. In addition to the motion estimation kernel trackers also take target appearance into account by extracting features from the target region, such as a bounding box. Classical examples are the Mean-Shift tracker (MST) [383] and the Kanade-Lukas-Tomasi tracker (KLT) [384]. Silhouette trackers also track an object by means of an appearance and a motion model, however they do not only match a fixed-size kernel (bounding box), but instead estimate the target shape, for example the target contours, in every new frame. Classical examples are state space models [385] and the Hough transform [386].

Smeulders et al. [41] published another survey covering tracking algorithms published until 2011. Besides presenting and categorizing individual trackers, they also conducted an extensive performance comparison. They categorize existing trackers based on wether they use *template matching* or a *discriminative classifier* to locate the tracking target in a new frame. The first group comprises basic template matching with trackers, such as NCC [387], KAT [388] and FRT [389], template matching with an extended appearance model and IVT [390], TAG [391] and TST [392] trackers, and template matching of spare representations with trackers like TMC [393], ACT [394] and L1T [395]. The second group of trackers, which built a model for distinction between target and image background, is subdivided into basic discriminative trackers like FBT [396] and TLD [397], and discriminative trackers, which use a sparse representation of the target and background instances, such as STRuck (STR) [398].

Apart from these two surveys, the works of Cannons [40] and Maggio et al. [399] provide more fundamental introductions to the topic of object detection. Instead of a detailed presentation and comparison, they describe tracking applications, feature extraction methods, target appearance models, classification, ensemble fusion, multi-target tracking, context-aware tracking and common performance parameters.

The most recent and comprehensive overview on object tracking algorithms is the survey by Fiaz et al. [39]. Besides summarizing correlation filter and deep learning trackers, they also present recent algorithms, which built on the ideas of earlier classical trackers. They categorize those algorithms into *multiple instance learning trackers*, *superpixel-based trackers*, *graph-based trackers*, *part-based trackers*, *sparse trackers* and *patch learning trackers*. The last two categories are equal to the sparse discriminative trackers and template matching trackers in the survey by Smeulders et al. [41].

Multiple instance learning (MIL) [400] trackers utilize a discriminative classifier to distinguish between the target foreground and the image background, when classifying proposed future locations of the target in a new frame. However, instead of training the classifier on individually labeled samples of foreground and background patches, MIL trackers use bags of samples, which contain multiple background and target samples and have a single label for the entire bag. A bag, which contains only background samples is labeled as background, while a bag that contains at least one target sample is labeled as sample. Examples of MIL trackers are the original MIL tracker by Babenko et al. [401], an improved version called MILBoost [402], Patch Based MIL (PMIL) [403], Robust MIL [404], Yang's Mil [405], FMIL [406], CMIL [407] and Sharma's MIL [408].

Superpixel-based trackers use superpixels [409] as features to describe the target appearance and classify the region in which the target is located in a new frame. Superpixels are clusters of adjacent image pixels with similar colors and provide a robust solution for capturing the structure of objects in an image and distinguish between background and target. Superpixels are robust against occlusion and appearance changes of the target. Examples of superpixel trackers are the approach proposed by Jingjing et al. [410], an early algorithm by Wang et al. [411] and a more recent tracker by the same authors [412], the Constrained Superpixel Tracker (CST) [413], the Structural Superpixel Descriptor (SSD) [414] and the BacKGround tracker (BKG) [415].

Graph-based trackers model the tracking problem by building a graph [416], in which the nodes represent image parts, such as superpixels, local features and the edges represent spatial or temporal dependencies between these image parts. A broad range of graph based optimization strategies can then be utilized to either match image parts across subsequent images, to discriminate between tracking target and image background, or to locate the target in a new image. Examples for classical graph-based trackers are the Structure Aware Tracker (SAT) [417], the Graph Tracker (Gracker) [418], the Geometric hyperGraph Tracker (GGT) [419] and the Absorbing Markov Chain Tracker (AMCT) [420].

Part-based trackers do not model the tracking target as a holistic entity, but instead segment the target into several parts and model each of these parts separately. This improves tracking accuracy in case of heavy target occlusion and deformation. Examples for classical part based trackers are the Part-Based Multi-Graph Ranking tracker (PMGRT) [421], the Part-based tracker (PT) by Yao et al. [422], the Adaptive Local Movement Modeling (ALMM) tracker [423], the tracker by De Ath et al. [424] and the tracker by Li et al. [425].

Sparse trackers use sparse encodings [426] both as features that represent target and background image patches, and to search for the optimal target location in a new frame. A sparse encoding of an image patch is a linear combination of sparse coefficients and a set of dictionary items, which is learned in such a way, that they minimize the reconstruction error. Examples of classical sparse trackers are the Sparsity-based Collaborative Model tracker (SCM) [427], the Consistent Low-Rank Sparse Tracker (CLRST) [428], the Structural Sparse Tracker (SST) [429], the Context-Aware Exclusive Sparse Tracker (CEST) [430], the Hierarchical Sparse Tracker (HST) [431], the tracker proposed by Guo et al. [432] and the Structured Multi-Task Multi-View Tracker (SMTMVT) [433].

Patch learning trackers are simple tracking approaches utilizing a discriminative classifier to localize the tracking target in a new frame by determining, whether a proposed region in a new frame is the target or the image background. Most of the recent classifiers in this category make use of deep learning, however there are also more classical approaches, such as the Edge Box Tracker (EBT) [434], the Exemplar-based Linear Discriminant Analysis tracker (ELDA) [435] and the Oblique Random Forest tracker (Obli-Raf) [436].

B.4 Correlation Filter Based Tracking Algorithms

One large category of modern object tracking algorithms utilize correlation filters [12, 179, 437] to determine the target location of the tracking object in a new frame. The common pipeline of correlation filter trackers (CFTs), which can be found in [42] and [39], is very similar to the tracking pipeline presented above, however instead of utilizing a motion model to predict candidate locations of the target in a new frame based on the previous motion path of the target, CFTs perform an exhaustive search in the local neighborhood of the target location in the previous frame [42]. The classifier module in the above tracking pipeline is replaced by a correlation filter W, which is circularly convolved with the input features X, yielding a two-dimensional gaussian probability density map Y of the target location in the new frame. This can be written as

$$Y = W \circledast X. \tag{B.1}$$

The target position in the new frame can now easily be found by computing the maximum of this distribution. As described earlier, the input features X can be of various kinds, such as HOG features or color histograms, and are extracted from the image region around the target position in the previous frame. More recently, features are also extracted at various different scales by means of an image pyramid to increase robustness of the tracker [438–441]. As circular convolution $\circledast$ in the spatial domain can be expressed as element-wise multiplication $\odot$ in the fourier domain [442], the response map can be computed as

$$Y = \mathcal{F}^{-1}\left\{\hat{W} \odot \hat{X}\right\}, \tag{B.2}$$

where $\hat{W}$ and $\hat{X}$ denote the correlation filter and input image in fourier domain after application of the discrete fourier transform (DFT) [443] and $\mathcal{F}^{-1}$ is the inverse discrete fourier transform (iDFT) [444]. Replacing the convolution with the element-wise product in the fourier domain leads to a significant increase in processing speed [445]. Since the parameters of the correlation filter W are initially unknown, training is required [39]. Given ground truth response maps Y_L, the optimal filter W^* can be found by iteratively optimizing the following objective [446]

$$W^* = \arg\min_{W} ||W \circledast X - Y_L||^2. \tag{B.3}$$

The correlation filter can be seen as a linear discriminative classifier, that determines, whether a location in the input frame is the target or the image background. To increase accuracy of the filter and make use of high-dimensional input features, the linear filter has to be replaced with a non-linear filter, yielding a *kernelized correlation filter*. This is usually done by applying the *kernel trick* [447], which yields a non-linear filter in input variables Z while still being linear, and thus easy to train, in input variables X. In a kernelized correlation filter, the filter matrix is replaced by the *kernel matrix*. [179]

The literature on CFTs is vast [39, 42] as the filter approach to tracking has proven to be very accurate and fast. Literature [39] categorizes CFTs into *basic*, *regularized* and *part-based* CFT (see fig. B.1), thus the following overview is structured accordingly. Here, only CFTs, which do not utilize any deep features, are presented, as these will be covered in a later section.

B.4.1 Basic Correlation Filter Trackers

One of the first basic CFTs to follow the above described mathematical framework was the Average Synthetic Exact Filter (ASEF) [448]. A year later Minimum Output of Sum of Squared Error (MOSSE) [12] was proposed, which provided a solution for online-training of the correlation filter. An improvement of MOSSE was the Circulant Structure with Kernel tracker (CSK) [437], which exploitet the circulant structure of the kernel matrix in case of dense sampling of the input space to overcame the problem of a high computational cost of training a kernelized correlation filter with many input samples. Danelljan et al. [449] improved CSK by utilizing color attributes as additional input features and implementing an adaptive dimensionality reduction technique to increase processing speed. Henriques et al. [179] proposed the Kernelized Correlation Filter tracker (KCF), that builds on the idea of utilizing the kernel trick and circulant kernel matrices to reduce storage and computation footprint of the correlation filter. This paper also introduces the Dual Correlation Filter (DCF), which is a fast multi-channel version of linear correlation filters. The Spatio-Temporal Context tracker (STC) [450] is an early approach to track targets at different scales. Another scale-adaptive tracker, the Scale Adaptive with Multiple Features tracker (SAMF) [451] builts on KCF, but uses HOG and color features at various different scales to generate richer input features and increase overall tracking performance. An improved version of SAMF, which uses global context to achieve a gain tracking accuracy is presented in [452]. The Joint scale-spatial correlation tracking with adaptive rotation estimation (RAJSSC) tracker [453] improves the single-template appearance models of the early CFTs by modelling the tracking target simultaneously at different spatial displacements, scale variations and rotations. The same authors presented a further improvement of the RAJSSC tracker, called JSSC [454], which exploits the block-circulant structure of the kernel matrix in their proposed joint scale-space to speed up template matching. Discriminative Scale Space Tracker (DSST) [438] is another tracking algorithm, which aims at tracking targets through scale-changes by utilizing one correlation filter for location estimation and a second one for scale estimation. Another correlation filter based tracker is the MUlti-Store Tracker (MUSTer) [455], which is inspired by the Atkinson-Shiffrin Memory Model of memory in the human brain and models appearance of the tracking target with short- and long-term memories. Ma et al. [456] proposed the Long-term Correlation

Tracker (LCT), which is based on a correlation filter and aims at tracking targets, that undergo significant appearance changes, such as deformation, abrupt motion, heavy occlusion and out-of-view scenarios. The Sum of Template And Pixel-wise LEarners tracker (STAPLE) [457] uses illumination-invariant HOG features and deformation-invariant color features with two separate correlation filters and merges both filter responses to obtain the estimated target location. A recent correlation filter, which relies on HOG features is the Background-Aware Correlation Filter (BACF) [458], that learns apperance models for both the target as well as the image background to increase tracking performance.

B.4.2 Regularized Correlation Filter Trackers

Regularized CFTs are the second category of CFTs [39]. The presented basic CFTs share the common boundary condition of equally sized filter matrices W and input features X of image patches in the local neighborhood of the last known target position. Since these are rectangular shapes, it is not always guaranteed that the learned filter matrix and the tracking target align well. Instead, the filter might also take parts of the image background around the target into account, degrading tracking performance. Regularized CFTs solve this problem by assigning learnable regularization parameters to the correlation filter, which penalize their spatial location and thus allow the filter to better fit to the target shape and ignore the background around the target. Danelljan et al. [459] laid the foundations for regularized CFTs and introduced the Spatially Regularized Discriminative Correlation Filter (SRDCF) for tracking. A more recent regularized CFT is the Discriminative Correlation Filter Tracker with Channel and Spatial Reliability (CSRT) [13], that uses HOG and color features and utilizes binary parameters to classify, which parts of the filter belongs to the tracking target and which ones to the image background.

B.4.3 Part-Based Correlation Filter Trackers

The third big group of CFTs are part-based CFTs [39], which break down tracking of a bounding box around the target into tracking of individual target parts. This makes the tracker more robust to occlusion and deformation of the target. Instead of learning one filter matrix for the entire object, part-based trackers learn multiple filter matrices, for the individual parts of the target. Examples are Real-time Part-based tracking with Adaptive Correlation Filters (RPAC) [460] and the Real-time Part-based tracking with Discriminative Correlation Filters (RPAC+) [461] algorithms, which add additional spatial constraints to the individual part filters and are robust against various appearance changes of the target. Other part-based trackers are the Reliable Patch Tracker (RPT) [462], Patch-based KCF [463], the Enhanced Structural Correlation tracker (ESC) [464] and the Structural Correlation Filter tracker (SCF) [465].

B.5 Deep Learning Based Tracking Algorithms

More recent approaches to object tracking are mostly based on deep learning and are covered in the surveys [38, 466, 467] and [39]. As shown in fig. B.1 and according to [38], deep learning trackers can be categorized by their level of integration of the tracking pipeline steps (see section 3.2). A large group of deep learning trackers utilize a deep architecture to extract robust and meaningful deep features of the image patches and feed those into correlation filter based trackers [15, 468–471]. Hence, this category of deep learning trackers can be sub-divided similar to appendix B.4 into basic deep CFTs, regularized deep CFTs and part-based deep CFTs. Instead of using a correlation filter, some deep learning trackers feed the extracted deep features into a more classical tracking pipeline and perform deep patch learning [472–476]. Another large group of tracker model the entire tracking pipeline in a single deep architecture, which can be trained in an end-to-end fashion. This category can be further subdivided into Siamese trackers [14, 46, 477–482], CNN-based trackers [483–487], RNN-based trackers [488–490] and trackers making use of reinforcement learning [491–495]. In case of multi-tracking, some trackers utilize a deep architecture as target associator, that matches multiple targets across frames [46–48], and another family of deep trackers makes use of deep models for the motion prediction step of the pipeline [496, 497]. The following section presents a variety of deep learning based tracking algorithms of the mentioned categories in more detail.

B.5.1 Deep Learning for Feature Extraction

Deep learning provides several types of models, which are used to extract meaningful and robust features from images and image sequences and thus can be used to represent the tracking target in the object tracker. The majority of deep feature extractors are based on convolutional neural networks (CNN) [498, 499], which take an image as input and convolve a weight kernel with the image pixels in a sliding window fashion to retrieve a reduced representation of the input image [288]. Some examples for trackers using CNNs for feature extraction are SO-DLT [500], the work by Hong et al. [476], the work by Jin et al. [501], MDNet [502] and the hedged deep tracker by Qi et al. [445]. Ma et al. [503] provide an in-depth analysis of deep convolutional features for object tracking and analyse the importance of individual feature maps of a multi-layer CNN for the tracking result. A slight variation of the standard CNN are deep residual features [177], which are used by the CREST tracker presented in [446]. Apart from convolutional models, recurrent neural networks (RNN) [504] are used for feature extraction in object trackers. RNNs are used to encode temporal or spatial sequences, such as the varying location of the tracking target over time [505] or spatial positions of image pixels [506, 507]. For object tracking it is also common to combine both CNN and RNN, yielding a Recurrent-CNN, which first extracts spatial features of every individual frame with a CNN and then encodes the spatial features of multiple subsequent frames with an RNN. Examples for this approach are Track-RNN [508] or the RFL network [509]. A similar approach is presented in [510]. Here no RNN is used, but instead a CNN encodes both spatial and temporal information about the tracking target. Yet another deep learning model, used for feature extraction in object tracking, is the stacked denoising autoencoder (SDAE) [511], which is a combination of a multi-layer encoder and decoder, which first compresses the input image into a reduced feature set and then decompresses these features back to an image similar to the input image. The encoder network can be used as a feature extractor for object tracking. Example trackers utilizing an SDAE for feature extraction are the DLT algorithm [472], the tracker by Hua et al. [512] and the algorithm by Zhou et al. [513].

Ma et al. [468] proposed CF2, a basic deep CFT, that uses three differently scaled CNN feature maps from VGG-19 [288], trained offline on ImageNet [295], as input features for three independent adaptive correlation filters. The peak probabilities in all filters are then jointly used as an estimate for the new target location. Model update is performed online on the correlations filters, while the CNN serves only as a pre-trained feature extractor with fixed weights. The authors show the superiority of deep features compared to hand-crafted features in terms of robustness to appearance changes of the target. The same authors also proposed the Hierarchical Correlation Feature-based Tracker (HCFT*) [503], which uses an additional correlation filter for scale-estimation and re-detection of targets. Similar to CF2, the Hedged Deep Tracker (HDT) [445] follows a similar approach and utilizes an individual correlation filter for each of six feature maps of a pre-trained VGG-19 to create a weak tracker. An adaptive hedging algorithm [514] is then used to combine all weak trackers into a strong tracker. The Multi-task Correlation Particle Filter tracker (MCPF) [471] has a similar architecture like CF2 and HDT, but proposes a method to jointly learn correlation filters and thus exploit inter-dependencies between the extracted feature maps. Additionally, a particle filter is used as motion model to generate proposals for the search regions in a new frame. Deep Large Margin Object Tracking with Circulant Feature Maps (DeepLMCF) [470] uses three feature maps extracted by VGG-19 in combination with a structured output SVM as classifier, which directly estimates the relative movement of the target between adjacent frames. Correlation filter theory is applied to speed up computations of the SVM. The Convolutional RESidual Tracker (CREST) [446] follows the standard framework of a discriminative correlation filter tracker, but reformulates the correlation filter via a combination of a single-layer base CNN and additional residual layers. All layers are trained fully online to adopt to the tracking target, thus no pretraining is needed. The Multi-Cue Correlation Filter tracker (MCCT) [515] maintains an ensemble of seven basic CFTs, which operate on different subsets of a feature pool, containing HOG-features, color-features and three different deep convolutional features extracted by a VGG-19 network.

An early example of a deep regularized CFT is the Deep Spatially Regularized Discriminative Correlation Filter tracker (deepSRDCF) [516], which is an extension of the previously presented SRDCF tracker, making use of five deep feature maps extracted by an ImageNet VGG-2048 CNN [517]. Their work finds, that deep features improve performance of the standard SRDCF tracker and that features of early layers provide superior tracking performance compared to deeper layers. Another deep regularized CFT, that uses SRDCF as a base tracker, is Deep Motion SRDCF [518]. In their framework, the authors fuse a set of hand-crafted and deep convolutional appearance features with convolutional feature maps extracted from an optical flow image [519] to capture both appearance as well as motion information of the target. The Continuous Convolutional Operators for Tracking algorithm (CCOT) [520] takes the image patch along with the feature maps of the first and last layer of a CNN as input and applies a set of correlation filters in a continuous range of resolutions. Outputs of the continuous correlation filters are combined into the final prediction of the new target location. The Efficient Convolution Operators tracker (ECO) [15] is an improvement of CCOT in terms of tracking accuracy and computational complexity. Compared to CCOT, ECO reduces the number of correlation filters, that need to be trained to estimate the target location, and furthermore uses a Gaussian Mixture Model to represent target appearance making ECO less prone to overfitting. Another deep regularized CF is the Attentional Correlation Filter Network tracker (ACFN) [521], which uses an attention network to adaptively select a subset of correlation filters out of a pool of KCFs, based on the motion properties of the target.

The last subcategory of deep correlation filter trackers are deep part-based CFTs, which follow the approach of tracking the target not as a whole, but instead as individual parts of the target. An example is the Recurrently Target-Attending Tracking algorithm (RTT) [522], which first partitions the target into a grid of parts, extracts HOG- and CNN-features of the parts and feeds them into a quad-directional RNN to generate a confidence map for the importance of every single part for the tracking task based on the learned spatial relationships between parts. The confidence map is then used to regularize a discriminative correlation filter, that estimates the final target location.

Instead of using correlation filters for location estimation of the target based on extracted deep features, some deep learning trackers feed deep features into more classical frameworks to estimate the target location. An early example is the Deep Learning Tracker (DLT) [472], which uses the encoder part of an offline trained autoencoder to extract appearance features from candidate regions, which are predicted via a particle filter motion model. During tracking, the encoder is extended with a sigmoid layer, which classifies each candidate patch as image background or target. Hong et al. [476] propose a tracker based on a pretrained CNN for deep feature extraction of sampled candidate regions. A SVM is used for discrimination between target and background patches. Additionally, target patches are used to compute a saliency map [523] via a backward pass through the CNN. The saliency map is convolved with a generative appearance model, that is built from saliency maps of previous frames, and outputs a dense likelihood map for the target location. Both SVM weights as well as the generative apperance model are updated after every frame. The Convolutional Networks without Training tracker (CNT) [473] uses one convolutional layer to extract features from densely sampled and normalized patches around the target in two subsequent frames and convolves them in a second convolutional layer to produce the final feature tensor. Both convolutional layers are trained online to adopt to appearance changes of the target. Tracking in CNT is performed by a classical particle filter framework. The Deep Relative Tracker (DRT)

[474] also utilizes a particle filter framework to propose candidate locations of the target in a new frame. Instead of directly extracting features and determining whether a patch is the image background or the target, DRT utilizes a deep CNN and a newly proposed relative loss layer to generate an overlap score for pairs of image patches, and thus model relative instead of absolute appearance information of candidate regions. The network consists of two parallel branches of five convolutional and five fully-connected layers with shared weights, and is trained on multiple pairs of image patches and the corresponding overlap score. During tracking a single candidate patch is fed into one of the network branches and an overlap score with the target template is returned. The candidate region with maximum score represents the new target location. The deep relative network is updated online based on the overlap scores of the candidate regions. A tracker, which makes use of deep appearance models in combination with a classical tree-based tracking approach is the Tree structure CNN tracker (TCCN) [475]. Here, candidate regions are randomly sampled from the region around the last known target location, and features are extracted with multiple different CNNs. Each CNN represents a node in a graph and has a scalar weight, that determines the importance of the corresponding CNN features for the final classification. Graph edges model relations between the CNNs and are needed for onloine updating of the individual CNNs.

B.5.2 End-to-End Deep Learning Models

Another category of deep learning trackers integrate all steps of the tracking pipeline in a single deep neural network, making them end-to-end trainable. A sub-group of these end-to-end trainable deep trackers are Siamese neural networks [524–526], which take two images as input and output a similarity score for those two images. This score can be used to classify, in which of several proposed image patches in the next frame the target is located by simply computing the similarity between an image of the target and image patches of the proposed future locations. Siamese Instance Search for Tracking (SINT) [46] was the first tracker to follow this approach. It was trained completely offline, but delivered state-of-the-art performance in 2016. Another offline trained Siamese neural network tracker is the Generic Object Tracking Using Regression Network (GOTURN) [479], which uses the first five layers of the CaffeNet architecture [527] to extract features from the target template and the query frame and compares them by means of four fully connected layers, which finally output bounding box coordinates of the target in the query frame. Another relatively simple tracker based on a Siamese neural network is the Fully Convolutional Siamese Network tracker (SiameseFC) [477], which embeds images of both the tracking target as well as a much larger region in the query frame into a lower-dimensional feature space. Both features are then combined with a cross-correlation-layer, yielding a spatially supported similarity map for the target location in the query frame. The cross-correlation-layer is nothing else than a correlation filter with features of one image as input and features of the other images as filter matrix. The search is repeated with differently scaled query frames to enable tracking through different scales. Despite not incorporate any appearance model, additional features or bounding box regression for refinement of the estimated bounding box, SiameseFC achieved state-of-the-art performance in 2016. The same authors proposed the Correlation Filter Network (CFNet) [481], which is an enhancement of SiameseFC with a similar architecture. The only change, introduced in CFNet is an additional correlation filter layer in the network branch of the target image, which serves as an online trainable appearance model of the tracking target. Another improvement of CFNet is the Dynamic Siamese Network tracker (DSiam) [478], which extends CFNet by an additional online trainable correlation filter in the network branch of the query frame. This correlation filter is used as regularizer and adaptively learns background suppression from previous frames. In combination with the correlation filter in the target branch, which models target appearance from previous frames, DSiam effectively tracks targets even under difficult conditions, such as fast appearance changes, occlusion and background clutter. The Discriminative Correlation Filters Network tracker (DCFNet) [14] works similarly to CFNet, however here the trainable correlation filter layer is not located in the target template branch, but instead combines features from both branches.

Hyperparameters of the correlation filter layer are trained jointly with the parameters of the convolutional Siamese feature extractors by backpropagation in an offline training step prior to tracking. During tracking only parameters of the correlation filter are updated to preserve a model of target appearance in previous frames. Vondrick et al. [482] train a Siamese CNN to copy colors from the first colorized frame of a video (reference frame) onto subsequent gray-scale frames (target frames). This model implicitly yields a soft pointer from pixels in a frame to the corresponding pixels in the next frame and thus can be exploited for object tracking. The Quadruplet-CNN tracker (Quad-CNN) [480] is not a classical Siamese neural network, because it takes four instead of only two input patches and ranks them based on their similarity in both the extracted appearance features as well as their temporal distance. Additionally, the network, which is trained offline and end-to-end, performs bounding box regression.

Apart from Siamese trackers, there are further deep trackers, which comprise the entire tracking pipeline in a single deep neural network. An example is DeepTrack [483, 484], which uses two convolutional and two fully-connected layers to extract robust appearance features from 1500 randomly sampled patches around the previous location of the target and classify them as background or target. The CNN is trained online on the found target via lazy update, which means training occurs only in case the training loss exceeds a defined threshold. The Multi-Domain Network tracker (MDNet) [502] utilizes a unique architecture of three convolutional and two fully-connected layers, which branch out into multiple domain-specific fully-connected layers. During offline training each domain-specific layer is trained separately on videos containing only a certain object category of tracking targets. This way, the previous shared layers learn a generic appearance model of all targets. During tracking the domain-specific layers are replaced with an uninitialized fully-connected layer, which is trained online on the current tracking target and thus stores target-specific information across subsequent frames. Two other deep end-to-end trackers, which are trained in a similar way as MDNet, are the the BranchOut tracker [528] and the Structure Aware Network tracker (SANet) [506]. BranchOut uses the MDNet base architecture followed by multiple branches with either one or two fully-connected layers. Different from MDNet, BranchOut maintains the pool of parallel branches even during tracking and updates a randomly sampled subset of branches after every tracking step. This serves as a regularization technique to maintain a robust appearance model of the target. SANet extends the MDNet architecture by three additional RNN-layers behind each convolutional layer, which capture spatial structure of the entire image, and concatenation units, which skip the RNN-layers and concatenate the CNN features with the RNN features. The Sequentially Training Convolutional Tracker (STCT) [487] uses the first ten convolutional layers of a VGG-16 network, that was pretrained offline, and interprets every channel of the CNN output as an individual weak tracker generating an individual confidence map of the target location in a new frame. Combining confidence maps of all weak learners in the ensemble yields robust tracking results. After determination of the new target location as maximum of the combined confidence maps, a scale prediction network estimates the new target scale and parameters of the feature extraction CNN are updated online. The Temporal Spatial Network tracker (TSN) [486] uses three deep networks, a convolutional feature network, a fully connected temporal network and a convolutional spatial network, to locate the target in each video frame. The feature CNN provides general deep features of the target, which are used in the fully-connected temporal network to encode the target trajectory. In the temporal network an additional tuple learning module is used to store and utilize important historic samples. The spatial CNN uses the output of the temporal network to further refine the target location. Sun et al. [485] introduce the Spatial-Aware Regressions Tracker (LSART), which uses a novel spatial-aware kernelized ridge regression network to capture holistic features of the target, and a spatial-aware CNN, which focuses on smaller localized regions. LSART provided the most accurate tracking results of all trackers in the Visual Object Tracking Challenge (VOT) in 2017 [529].

Other deep end-to-end trackers are mainly based on RNNs to model temporal characterstics of the target. For example, the multi-target tracker by Milan et al. [489] uses an individual RNN to model the motion and predict the location of each target, and a single LSTM for target association. The entire model is trained offline in an end-to-end fashion. Similarly, the tracker by Gan et al. [488] is trained offline as an end-to-end model. It uses a RNN, which directly outputs bounding box coordinates based on the previous hidden state and CNN-features extracted from the current video frame. The Recurrent YOLO tracker (ROLO) [490] is another tracker, which directly regresses bounding box coordinates for every frame in a sequence. It applies the YOLO object detector to every frame of the video and feeds the found bounding boxes into a LSTM, which maintains a memory of the detections in previous frames. This yields bounding box coordinates, which are not only based on the detections in the current frame, but also on the detections in previous frames. Thus, ROLO is capable to model both spatial as well as temporal dependencies of targets across frames.

Deep reinforcement learning (RL) [530] provides another class of end-to-end trainable object trackers and models the tracking problem as a decision-making process with the goal to maximize a reward function. RL systems comprise an environment with a state $s \in \mathbb{S}$, and an agent, which observes the state and chooses an action $a \in \mathbb{A}$ based on a policy function $\pi(s) : \mathbb{S} \rightarrow \mathbb{A}$. Depending on the state and the chosen action, the agent receives a reward $R(s, a) \in \mathbb{R}$, which is used to find the optimal policy during agent training. When the optimal policy is found, it is guaranteed, that the agent always decides for that action, that maximizes the accumulated reward. A tracker following this approach is the Action-Decision Network tracker (ADNet) [494], which uses a three convolutional and three fully-connected layers as an agent network. Given a new frame, in which the target has to be located, ADNet iteratively outputs an action to shift or rescale the target's bounding box from its current to a new location until the target has been found. The reward used for offline training of ADNet is based on the IoU (see appendix A.3) between the predicted and the ground-truth bounding boxes in the training dataset. The fully connected layers are retrained in an online manner during tracking to better capture appearance changes of the target. ADNet significantly speeds up the tracking process compared to non-RL trackers as only a low number of iterations is needed to find the new target location. The Actor-Critic Tracker (ActNet) [493] is based on a very similar architecture, but instead of learning the optimal policy function offline, it makes use of the actor-critic framework [530, 531] to find the optimal policy online during the tracking process. The Reinforced Decision Making tracker (RDM) [491] comprises of a Siamese CNN matching network and a convolutional policy network. The Siamese matching network takes a query image and a target template as inputs and generates a probability map for the spatial location of the template in the query image. This prediction map is fed into the policy network, which computes a normalized score for each probability map of a template drawn from a pool of templates, and then selects the template, which leads to the highest normalized score, for localizing the target. The EArly Stopping Tracker (EAST) [492] is also based on the typical Siamese CNN framework for computing target location in a new frame. Here, an RL agent decides, how many CNN layers are used to compute feature maps for location estimation. In case of easily distinctive and slow moving targets, the agent uses only a few layers, while in more challenging tracking situations, deeper features are needed, and thus more CNN layers are utilized. This early-stopping approach leads to a significant speedup of the feedforward computation and hence the entire tracking process. The Deep Reinforcement Learning Tracker (DRLT) [495] estimates the target location in every frame from the according hidden state of a RNN. The hidden state for a given time step is computed from the previous hidden state and a feature vector extracted from the current video frame. During offline training the RL agent receives a reward based upon the accuracy of the predicted target location. Aim of the training is to iteratively update the parameters of the policy function to maximize the accumulated reward. Parameters of the policy function are equal to the parameters of the RNN and the feature extraction CNN, which means that finding the optimal policy function is equal to finding optimal parameters for the CNN and RNN.

B.5.3 Deep Learning for Target Association

In case of multi-target tracking deep learning can be explicitly utilized for the task of target association, that means matching different tracking targets in adjacent frames. Siamese neural networks provide a reasonable solution for this task [46–48]. Leal-Taixe et al. [47] proposed a Siamese CNN architecture to match detections of multiple pedestrians across frames. Local spatio-temporal features of each target region are extracted via a pre-trained Siamese CNN and fed into a gradient boosting classifier together with additional contextual features to generate output matches between targets in adjacent frames. Linear programming in then used for offline generation of trajectories. A second example for target association via deep learning is the Siamese LSTM tracker proposed by Varior et al. [48]. In both branches of the Siamese network, hand-crafted features are extracted from several horizontal patches of the target and the query image and fed into interconnected LSTM cells. A contrastive loss function is used to merge both branches of the LSTM and generate the final prediction of the target location in the query frame. The LSTM allows the network to propagate spatial context between multiple target and thus helps to match them across frames.

B.5.4 Deep Learning for Motion Prediction

The last category of deep tracking algorithms relies on deep learning solely for motion prediction. An example is the Behavior-CNN proposed by Li et al. [496], which is based on the KLT tracker [384], however replaces its motion prediction by a CNN that generates trajectories of humans in crowded scenes and thus can be applied to predict an estimate of the location of a human in the next frame. Similarly, the Social-LSTM [497] by Alahi et al. utilizes an individual LSTM for every single person in the frame to predict the future location in the next frame.

Appendix C

Appendix

This appendix contains additional material. First, the hard- and software specification of the workstation, on which the Shopfloor Monitor is executed, is presented. Afterwards, the full results of experiments, conducted in chapter chapter 6, are listed. Finally, the datasheet of the Shopfloor Monitor's two cameras is shown.

C.1 Server Hardware and Software

The Shopfloor Monitor is developed and executed on a single desktop workstation with the hardware specs as listed in table C.1 and Ubuntu 18.04.1 LTS (64 bit) as operating system. Python 3.6.5 as part of Anaconda 5.2.0 is used to implement most parts of the monitoring system. Additional Python packages, which are not part of Anaconda, such as Flask 1.0.2 or Flask-SocketIO 3.0.1, are installed as needed and managed in a virtual environment created with the VENV module of the Python standard library. MySQL Server 5.7.23 and Redis 4.0 are installed directly in the operating system and configured as services. For GPU inference of the object detector, Tensorflow-GPU 1.8.0 is installed via Python pip, after setup of CUDA 9.0 and cuDNN 7.1.

Tab. C.1: Hardware specification of the analysis server.

Component	**Type**	**Description**
CPU	Intel Core i7-8700K	3.70 GHz (4.70 GHz Boost), 6 Cores, 12 Threads, 12 MB Cache, 95 W TDP
GPU 0 / 1	Zotac GeForce GTX 1080 TI Mini	11 GB GDDR5X VRAM, 1506 MHz (1620 MHz Boost), 3584 Cuda cores
RAM	Corsair Vengeance LPX	4× 16 GB DDR4, 3200 MHz (PC4 25600)
Mainboard	MSI Z370 Gaming Pro Carbon	Socket 1151, Intel Z370 chipset, 3× PCI-E 3.0 x16
SSD	Samsung 970 EVO M.2 NVMe 1.3	1000 GB, Seq. read: 3500 MB/s, Seq. write 2500 MB/s, 1 GB Cache
PSU	Corsair RM1000x Gold	1000 W, 80 PLUS Gold
Case	Phantek Eclipse P400S	–

C.2 Full Results of Experimental Analysis

This section contains the full results of the experiments, conducted in chapter 6. Every subsection shows a table for each analyzed parameter value. For example, appendix C.2.1 contains a table for each of the seven tested object detection algorithms. Each table contains the MOT evaluation metrics as measured on the five evaluation sequences. The last rows, which were shown earlier in table 6.5, aggregate the metrics over all sequences.

C.2.1 Full Results for Detection Algorithms

Tab. C.2: Results for the Faster R-CNN/NASNet detector. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	77.2	78.9	75.5	86.4	90.4	18	14	2	2	330	485	9	13	77.0	18.6	38.51
Seq-1	61.3	66.7	56.7	70.0	82.5	13	5	8	0	222	447	9	15	54.5	19.7	60.10
Seq-2	66.6	78.8	57.7	65.2	89.0	19	7	10	2	234	1014	14	17	56.7	20.4	58.83
Seq-3	71.5	78.7	65.6	71.7	86.0	17	7	10	0	328	798	10	18	59.7	20.3	43.75
Seq-4	63.0	69.5	57.6	73.4	88.4	10	3	5	2	134	371	8	15	63.2	15.0	64.77
Total	70.0	76.3	64.6	74.5	87.9	77	36	35	6	1248	3115	50	78	63.8	19.0	53.19

Tab. C.3: Results for the Faster R-CNN/Inception-ResNet v2 detector. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	84.4	91.0	78.7	82.5	95.4	18	13	4	1	143	624	3	24	78.5	20.6	42.54
Seq-1	70.3	78.9	63.3	71.8	89.5	13	5	7	1	125	420	8	13	62.9	19.7	60.60
Seq-2	66.0	78.2	57.0	66.8	91.6	19	7	9	3	179	968	11	17	60.2	20.7	65.37
Seq-3	75.2	87.4	66.0	70.2	93.1	17	8	9	0	147	839	6	18	64.8	21.0	44.07
Seq-4	61.8	67.5	56.9	73.7	87.5	10	4	4	2	147	366	8	14	62.6	15.2	68.34
Total	73.7	83.1	66.2	73.6	92.4	77	37	33	7	741	3217	36	86	67.2	20.0	56.18

Tab. C.4: Results for the Faster R-CNN/ResNet-101 detector. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	84.3	90.0	79.2	81.3	92.4	18	13	4	1	238	669	2	17	74.6	21.1	44.71
Seq-1	70.7	77.5	65.1	70.2	83.5	13	5	7	1	206	445	5	17	56.0	23.5	63.49
Seq-2	69.4	80.6	60.9	66.4	87.8	19	7	10	2	268	980	10	13	56.8	23.1	64.28
Seq-3	72.0	83.4	63.3	68.2	89.8	17	6	10	1	219	897	7	27	60.2	23.7	45.86
Seq-4	58.6	62.8	54.8	75.5	86.5	10	5	3	2	164	341	9	19	63.1	18.0	72.03
Total	73.4	81.6	66.7	72.7	89.0	77	36	34	7	1095	3332	33	93	63.4	22.0	58.08

Tab. C.5: Results for the Faster R-CNN/ResNet-101 (low proposals) detector. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	76.3	86.0	68.6	75.6	94.9	18	9	5	4	146	871	6	20	71.4	20.9	45.13
Seq-1	65.6	77.7	56.8	60.9	83.3	13	4	7	2	182	583	4	20	48.4	22.9	66.57
Seq-2	60.0	76.4	49.4	59.8	92.6	19	3	12	4	140	1172	13	21	54.5	23.2	71.66
Seq-3	65.8	81.4	55.2	62.2	91.6	17	5	11	1	160	1066	7	31	56.3	23.9	51.52
Seq-4	63.7	69.0	59.1	76.4	89.3	10	5	3	2	128	329	7	14	66.7	18.5	73.07
Total	67.4	79.7	58.4	67.0	91.5	77	26	38	13	756	4021	37	106	60.5	21.9	61.59

Tab. C.6: Results for the Faster R-CNN/ResNet-50 detector. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	77.5	84.7	71.4	77.4	91.8	18	12	3	3	246	809	7	25	70.3	24.1	44.16
Seq-1	59.9	71.9	51.2	61.1	85.8	13	3	9	1	151	580	6	15	50.6	24.5	60.74
Seq-2	57.2	72.7	47.1	56.2	86.7	19	4	10	5	252	1277	12	16	47.1	25.0	67.76
Seq-3	68.5	81.6	59.0	64.9	89.7	17	6	9	2	210	990	11	32	57.1	24.5	46.24
Seq-4	58.5	63.5	54.3	76.2	89.2	10	5	3	2	128	331	11	18	66.3	18.1	73.09
Total	66.5	77.3	58.3	67.3	89.3	77	30	34	13	987	3987	47	106	58.8	23.6	58.40

Tab. C.7: Results for the R-FCN/ResNet-101 detector. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	71.7	71.0	72.3	78.3	76.8	18	12	4	2	844	777	7	27	54.5	23.4	35.05
Seq-1	65.1	74.1	58.1	68.3	87.2	13	5	7	1	149	472	7	21	57.9	23.7	55.56
Seq-2	61.9	72.6	54.0	60.5	81.3	19	7	6	6	405	1152	10	23	46.2	26.0	59.71
Seq-3	64.7	72.2	58.6	63.8	78.6	17	6	10	1	490	1022	11	40	46.0	24.3	40.35
Seq-4	66.1	72.6	60.7	77.5	92.5	10	6	2	2	87	314	4	17	70.9	18.8	72.76
Total	66.5	72.1	61.7	69.3	81.1	77	36	29	12	1975	3737	39	128	52.8	23.6	52.68

Tab. C.8: Results for the SSD/ResNet-50 detector. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	66.7	73.8	60.8	72.9	88.6	18	9	6	3	337	968	10	16	63.2	21.5	44.53
Seq-1	54.3	69.6	44.5	53.5	83.8	13	3	7	3	154	693	7	7	42.7	22.7	65.28
Seq-2	44.7	73.4	32.2	38.9	88.6	19	4	7	8	145	1781	8	19	33.6	21.7	77.02
Seq-3	55.3	66.5	47.3	56.2	78.9	17	5	8	4	423	1235	14	22	40.7	24.0	43.17
Seq-4	52.7	64.7	44.5	61.5	89.5	10	1	7	2	101	536	6	15	53.8	20.9	76.20
Total	56.4	70.4	47.0	57.2	85.7	77	22	35	20	1160	5213	45	79	47.4	22.2	61.24

C.2.2 Full Results for Tracking Algorithms

Tab. C.9: Results for the MOSSE tracker. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	84.3	90.0	79.2	81.3	92.4	18	13	4	1	238	669	2	17	74.6	21.1	45.01
Seq-1	70.7	77.5	65.1	70.2	83.5	13	5	7	1	206	445	5	17	56.0	23.5	61.38
Seq-2	69.4	80.6	60.9	66.4	87.8	19	7	10	2	268	980	10	13	56.8	23.1	67.28
Seq-3	72.0	83.4	63.3	68.2	89.8	17	6	10	1	219	897	7	27	60.2	23.7	45.46
Seq-4	58.6	62.8	54.8	75.5	86.5	10	5	3	2	164	341	9	19	63.1	18.0	71.79
Total	73.4	81.6	66.7	72.7	89.0	77	36	34	7	1095	3332	33	93	63.4	22.0	58.18

Tab. C.10: Results for the KCF tracker. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	84.7	90.7	79.5	82.2	93.9	18	13	4	1	192	635	2	15	76.8	20.1	9.21
Seq-1	70.7	76.3	65.8	73.5	85.2	13	7	5	1	190	395	8	16	60.2	22.0	9.98
Seq-2	65.0	74.8	57.5	67.2	87.4	19	7	10	2	282	956	11	21	57.1	23.6	9.81
Seq-3	70.0	81.4	61.5	66.3	87.9	17	7	8	2	258	950	6	27	57.0	20.6	8.31
Seq-4	67.6	72.4	63.3	78.1	89.3	10	6	2	2	130	305	6	12	68.3	15.5	14.71
Total	73.1	81.1	66.5	73.4	89.5	77	40	29	8	1052	3241	33	91	64.5	20.6	10.40

Tab. C.11: Results for the CSRT tracker. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	83.0	88.3	78.3	83.0	93.6	18	14	3	1	202	607	4	21	77.3	20.1	2.68
Seq-1	71.1	77.1	66.1	72.9	85.1	13	6	6	1	191	404	7	19	59.6	21.6	3.32
Seq-2	66.6	75.6	59.5	68.6	87.2	19	8	9	2	294	916	13	20	58.0	23.8	2.92
Seq-3	72.9	84.0	64.4	69.4	90.5	17	7	9	1	206	864	6	29	61.8	20.4	2.88
Seq-4	67.3	72.0	63.1	79.5	90.8	10	6	2	2	112	285	6	11	71.1	15.1	4.89
Total	73.6	81.1	67.3	74.8	90.1	77	41	29	7	1005	3076	36	100	66.2	20.6	3.34

C.2.3 Full Results for Detector Cycle Frequencies

Tab. C.12: Results for the detector cycle frequency $f_d = 1$. Units of the metrics are according to table 6.3.

Sequence	IDF$_1$	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	83.4	87.6	79.5	85.0	93.7	18	14	2	2	205	536	7	20	79.1	20.9	7.81
Seq-1	69.3	73.2	65.8	77.7	86.5	13	7	6	0	181	332	10	28	64.9	23.7	9.03
Seq-2	65.1	73.6	58.4	70.0	88.2	19	8	9	2	272	875	21	27	59.9	23.4	9.27
Seq-3	70.4	78.4	63.9	74.3	91.2	17	8	8	1	202	724	18	50	66.5	22.9	8.21
Seq-4	73.9	78.1	70.2	84.5	93.9	10	7	1	2	76	216	8	20	78.5	17.5	9.84
Total	73.4	79.5	68.1	78.0	91.0	77	44	26	7	936	2683	64	145	69.8	21.8	8.83

Tab. C.13: Results for the detector cycle frequency $f_d = 2$. Units of the metrics are according to table 6.3.

Sequence	IDF$_1$	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	84.3	90.0	79.2	81.3	92.4	18	13	4	1	238	669	2	17	74.6	21.1	44.51
Seq-1	70.7	77.5	65.1	70.2	83.5	13	5	7	1	206	445	5	17	56.0	23.5	63.17
Seq-2	69.4	80.6	60.9	66.4	87.8	19	7	10	2	268	980	10	13	56.8	23.1	65.59
Seq-3	72.0	83.4	63.3	68.2	89.8	17	6	10	1	219	897	7	27	60.2	23.7	46.12
Seq-4	58.6	62.8	54.8	75.5	86.5	10	5	3	2	164	341	9	19	63.1	18.0	68.35
Total	73.4	81.6	66.7	72.7	89.0	77	36	34	7	1095	3332	33	93	63.4	22.0	57.55

Tab. C.14: Results for the detector cycle frequency $f_d = 5$. Units of the metrics are according to table 6.3.

Sequence	IDF$_1$	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	77.7	87.0	70.2	72.2	89.5	18	9	6	3	304	994	5	14	63.6	21.4	64.73
Seq-1	54.1	71.1	43.7	48.0	78.3	13	0	12	1	199	775	4	6	34.4	23.4	88.48
Seq-2	63.8	86.3	50.5	51.4	87.9	19	4	9	6	207	1415	3	6	44.2	24.1	97.84
Seq-3	51.0	65.6	41.7	47.5	74.7	17	3	10	4	454	1481	11	21	31.0	24.9	61.17
Seq-4	58.0	66.0	51.7	58.9	75.3	10	1	7	2	270	572	4	13	39.3	18.4	96.77
Total	63.4	77.8	53.6	57.0	82.9	77	17	44	16	1434	5237	27	60	45.1	22.5	81.80

Tab. C.15: Results for the detector cycle frequency $f_d = 10$. Units of the metrics are according to table 6.3.

Sequence	IDF$_1$	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	70.0	87.3	58.4	59.3	88.7	18	0	15	3	271	1456	4	10	51.6	21.9	70.27
Seq-1	37.7	84.2	24.3	24.5	85.1	13	0	5	8	64	1125	1	2	20.2	27.4	97.79
Seq-2	48.4	85.8	33.7	34.1	86.8	19	0	9	10	151	1920	1	3	28.9	24.6	105.63
Seq-3	38.1	68.3	26.4	28.2	72.8	17	0	8	9	296	2026	3	6	17.6	22.5	64.53
Seq-4	37.2	57.7	27.4	28.8	60.6	10	0	4	6	261	992	2	6	9.9	18.7	100.87
Total	50.9	79.7	37.4	38.3	81.8	77	0	41	36	1043	7519	11	27	29.7	22.8	87.82

C.2.4 Full Results for Frame Scaling Factors

Tab. C.16: Results for the scaling factor $s = 0.4$. Units of the metrics are according to table 6.3.

Sequence	IDF$_1$	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	84.3	90.0	79.2	81.3	92.4	18	13	4	1	238	669	2	17	74.6	21.1	43.76
Seq-1	70.7	77.5	65.1	70.2	83.5	13	5	7	1	206	445	5	17	56.0	23.5	61.95
Seq-2	69.4	80.6	60.9	66.4	87.8	19	7	10	2	268	980	10	13	56.8	23.1	67.26
Seq-3	72.0	83.4	63.3	68.2	89.8	17	6	10	1	219	897	7	27	60.2	23.7	46.68
Seq-4	58.6	62.8	54.8	75.5	86.5	10	5	3	2	164	341	9	19	63.1	18.0	74.55
Total	73.4	81.6	66.7	72.7	89.0	77	36	34	7	1095	3332	33	93	63.4	22.0	58.84

Tab. C.17: Results for the scaling factor $s = 0.6$. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	74.8	79.7	70.4	78.2	88.4	18	12	5	1	366	781	4	26	67.8	20.4	25.96
Seq-1	73.6	81.6	67.1	72.6	88.3	13	6	6	1	143	408	3	22	62.8	23.8	40.07
Seq-2	67.4	79.5	58.5	65.7	89.4	19	6	10	3	226	998	10	17	57.6	24.5	45.06
Seq-3	68.0	80.7	58.8	63.7	87.4	17	5	11	1	259	1025	5	30	54.3	23.6	27.64
Seq-4	56.6	62.9	51.5	71.2	86.9	10	3	5	2	150	401	9	13	59.8	17.4	47.78
Total	69.3	78.1	62.3	70.4	88.2	77	32	37	8	1144	3613	31	108	60.7	22.1	37.30

Tab. C.18: Results for the scaling factor $s = 1.0$. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	79.9	85.3	75.1	81.6	92.7	18	13	4	1	229	658	3	23	75.1	22.3	10.36
Seq-1	65.1	72.1	59.4	67.9	82.5	13	3	9	1	215	478	6	21	53.1	24.0	17.23
Seq-2	62.3	72.6	54.6	65.3	86.9	19	7	10	2	288	1010	12	24	55.0	24.4	21.81
Seq-3	66.5	80.5	56.6	62.7	89.2	17	5	11	1	215	1053	8	34	54.8	23.8	11.78
Seq-4	70.6	77.1	65.1	77.3	91.6	10	4	4	2	99	316	4	11	69.9	17.1	23.10
Total	69.9	78.8	62.9	71.2	89.2	77	32	38	7	1046	3515	33	113	62.3	22.6	16.85

C.2.5 Full Results for Match IoU Threshold

Tab. C.19: Results for the match IoU threshold $\theta_{\mathrm{IoU}} = 0.1$. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	87.0	92.9	81.9	82.0	93.1	18	13	4	1	217	642	0	17	76.0	21.1	45.67
Seq-1	70.2	75.0	65.9	73.8	84.0	13	5	7	1	209	390	9	19	59.2	23.4	62.88
Seq-2	67.8	77.2	60.4	68.2	87.2	19	7	8	4	292	926	12	21	57.8	23.1	68.00
Seq-3	77.3	88.6	68.6	69.7	90.1	17	6	10	1	216	854	6	29	61.8	23.5	47.81
Seq-4	68.0	71.8	64.5	77.2	86.0	10	6	2	2	175	318	7	21	64.1	18.0	73.02
Total	76.0	83.6	69.7	74.3	89.1	77	37	31	9	1109	3130	34	107	65.0	22.0	59.48

Tab. C.20: Results for the match IoU threshold $\theta_{\mathrm{IoU}} = 0.3$. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	84.3	90.0	79.2	81.3	92.4	18	13	4	1	238	669	2	17	74.6	21.1	44.56
Seq-1	70.7	77.5	65.1	70.2	83.5	13	5	7	1	206	445	5	17	56.0	23.5	62.05
Seq-2	69.4	80.6	60.9	66.4	87.8	19	7	10	2	268	980	10	13	56.8	23.1	66.52
Seq-3	72.0	83.4	63.3	68.2	89.8	17	6	10	1	219	897	7	27	60.2	23.7	47.25
Seq-4	58.6	62.8	54.8	75.5	86.5	10	5	3	2	164	341	9	19	63.1	18.0	69.23
Total	73.4	81.6	66.7	72.7	89.0	77	36	34	7	1095	3332	33	93	63.4	22.0	57.92

Tab. C.21: Results for the match IoU threshold $\theta_{\mathrm{IoU}} = 0.5$. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	76.4	82.1	71.4	79.8	91.7	18	11	6	1	258	723	8	14	72.3	21.1	44.82
Seq-1	60.7	67.1	55.5	68.5	82.9	13	4	9	0	211	470	11	18	53.6	23.3	61.57
Seq-2	64.3	76.7	55.4	63.7	88.2	19	6	11	2	249	1058	12	10	54.7	23.3	63.43
Seq-3	66.3	79.8	56.6	64.4	90.8	17	6	10	1	183	1003	9	22	57.6	23.4	46.03
Seq-4	50.4	56.4	45.6	68.7	85.0	10	2	6	2	169	436	13	17	55.6	18.7	67.31
Total	66.4	75.5	59.3	69.7	88.8	77	29	42	6	1070	3690	53	81	60.5	22.1	56.63

Tab. C.22: Results for the match IoU threshold $\theta_{\text{IoU}} = 0.7$. Units of the metrics are according to table 6.3.

Sequence	IDF$_1$	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	63.5	71.8	57.0	73.4	92.5	18	9	6	3	214	951	17	20	66.9	21.7	41.23
Seq-1	52.1	67.3	42.5	55.1	87.2	13	3	9	1	121	670	15	17	45.9	23.5	59.03
Seq-2	55.7	78.1	43.3	50.3	90.8	19	6	7	6	149	1449	19	17	44.5	23.5	60.52
Seq-3	51.8	71.9	40.5	53.0	94.0	17	4	9	4	95	1325	15	16	49.1	23.9	40.99
Seq-4	51.7	64.0	43.4	61.2	90.1	10	2	6	2	94	541	16	15	53.3	19.3	64.29
Total	56.4	71.6	46.6	59.5	91.5	77	24	37	16	673	4936	82	85	53.3	22.4	53.21

Tab. C.23: Results for the match IoU threshold $\theta_{\text{IoU}} = 0.9$. Units of the metrics are according to table 6.3.

Sequence	IDF$_1$	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	17.8	41.2	11.4	25.6	92.5	18	1	6	11	74	2661	27	28	22.7	20.8	31.31
Seq-1	3.4	72.2	1.7	2.4	100.0	13	0	0	13	0	1455	1	1	2.3	14.0	46.60
Seq-2	12.8	56.0	7.2	12.8	99.5	19	0	4	15	2	2540	8	8	12.5	18.6	50.63
Seq-3	11.0	70.6	6.0	8.1	95.8	17	0	2	15	10	2592	5	4	7.6	19.9	33.12
Seq-4	18.7	62.7	11.0	17.5	100.0	10	0	3	7	0	1149	6	5	17.1	14.1	52.18
Total	13.7	51.2	7.9	14.7	95.4	77	1	15	61	86	10397	47	46	13.6	19.2	42.77

C.2.6 Full Results for Hypothesis Thresholds

Tab. C.24: Results for the hypothesis thresholds $(0, 0)$. Units of the metrics are according to table 6.3.

Sequence	IDF$_1$	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	84.2	87.7	81.0	86.1	93.2	18	14	2	2	223	496	5	19	79.7	21.1	46.01
Seq-1	66.6	68.9	64.5	79.4	84.9	13	7	6	0	210	307	14	30	64.4	23.7	65.16
Seq-2	59.4	66.4	53.8	70.9	87.5	19	8	9	2	295	848	28	32	59.8	23.8	70.40
Seq-3	68.5	75.6	62.6	74.2	89.6	17	8	8	1	244	728	32	56	64.4	23.7	50.35
Seq-4	58.8	60.8	56.9	86.8	92.9	10	8	0	2	93	184	12	21	79.3	18.7	73.59
Total	69.7	74.6	65.5	79.0	90.0	77	45	25	7	1065	2563	91	158	69.5	22.3	61.10

Tab. C.25: Results for the hypothesis thresholds $(2, 2)$. Units of the metrics are according to table 6.3.

Sequence	IDF$_1$	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	86.0	90.2	82.1	84.8	93.1	18	14	2	2	224	543	4	18	78.4	21.1	45.25
Seq-1	68.3	71.2	65.6	78.9	85.6	13	7	6	0	198	315	12	25	64.8	24.1	62.73
Seq-2	64.3	72.2	58.0	70.6	87.9	19	10	7	2	284	855	19	24	60.2	23.7	69.41
Seq-3	73.3	81.6	66.5	73.9	90.8	17	8	8	1	211	735	15	45	65.9	23.8	46.25
Seq-4	57.4	59.6	55.3	83.7	90.2	10	7	1	2	126	227	13	21	73.7	18.4	73.12
Total	72.5	78.1	67.7	78.1	90.1	77	46	24	7	1043	2675	63	133	69.0	22.3	59.35

Tab. C.26: Results for the hypothesis thresholds $(5, 5)$. Units of the metrics are according to table 6.3.

Sequence	IDF$_1$	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	84.3	90.0	79.2	81.3	92.4	18	13	4	1	238	669	2	17	74.6	21.1	44.37
Seq-1	70.7	77.5	65.1	70.2	83.5	13	5	7	1	206	445	5	17	56.0	23.5	63.27
Seq-2	69.4	80.6	60.9	66.4	87.8	19	7	10	2	268	980	10	13	56.8	23.1	67.81
Seq-3	72.0	83.4	63.3	68.2	89.8	17	6	10	1	219	897	7	27	60.2	23.7	47.62
Seq-4	58.6	62.8	54.8	75.5	86.5	10	5	3	2	164	341	9	19	63.1	18.0	72.44
Total	73.4	81.6	66.7	72.7	89.0	77	36	34	7	1095	3332	33	93	63.4	22.0	59.10

Tab. C.27: Results for the hypothesis thresholds $(5, 0)$. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	83.6	91.0	77.3	79.9	94.1	18	13	3	2	179	718	3	19	74.8	21.1	45.76
Seq-1	69.0	83.6	58.8	63.0	89.6	13	4	9	0	109	552	10	25	55.0	22.8	65.64
Seq-2	56.2	75.1	44.9	56.6	94.6	19	5	12	2	95	1264	17	23	52.8	22.7	69.52
Seq-3	67.6	86.6	55.5	60.5	94.5	17	5	10	2	99	1113	14	33	56.5	23.6	50.49
Seq-4	58.5	67.9	51.4	72.4	95.6	10	4	4	2	46	385	11	21	68.3	18.2	74.51
Total	69.2	83.2	59.3	66.9	93.9	77	31	38	8	528	4032	55	121	62.1	21.8	61.19

Tab. C.28: Results for the hypothesis thresholds $(0, 5)$. Units of the metrics are according to table 6.3.

Sequence	IDF_1	IDP	IDR	RCLL	PRCN	GT	MT	PT	ML	FP	FN	IDSW	FM	MOTA	MOTP	FPS
Seq-0	83.5	85.5	81.7	86.5	90.4	18	14	3	1	327	484	5	19	77.2	21.2	44.61
Seq-1	67.4	65.3	69.7	81.8	76.7	13	8	5	0	371	272	15	25	55.9	24.3	59.05
Seq-2	69.0	70.2	67.9	75.1	77.7	19	10	7	2	629	725	16	22	53.0	24.0	65.73
Seq-3	71.1	71.3	70.9	80.7	81.2	17	11	6	0	527	543	19	40	61.4	24.0	46.37
Seq-4	58.5	56.4	60.7	87.7	81.6	10	8	0	2	276	171	12	20	67.0	18.9	70.96
Total	72.2	72.4	72.0	82.0	82.4	77	51	21	5	2130	2195	67	126	64.0	22.5	57.35

C.3 Camera Datasheets

This section contains the datasheet of the camera model used in the Shopfloor Monitor.

DH-SD22204T-GN

2Mp Full HD Network Mini PTZ Dome Camera

Features

- 4x optical zoom
- H.264 & MJPEG Triple-streams encoding
- Max. 25/30fps@1080P(1920×1080) & 50/60fps@720P resolution
- WDR, Day/Night(ICR), Ultra DNR, Auto iris, Auto focus
- Multiple network monitoring: Web viewer, CMS(DSS/PSS) & DMSS
- Max 100°/s pan speed
- Up to 300 presets, 5 auto scan, 8 tour, 5 pattern
- Support intelligent 3D positioning with DH-SD protocol
- Micro SD memory, POE

DH-SD22204T-GN

Technical Specifications

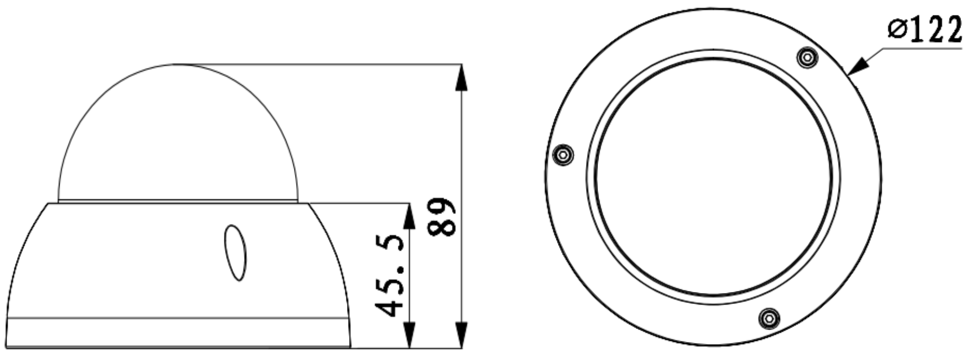
Model		DH-SD22204T-GN
Camera		
Image Sensor		1/2.7" CMOS
Effective Pixels		1920(H) x 1080(V), 2Megapixels
Scanning System		Progressive
Electronic Shutter Speed		1/1 ~ 1/30,000s
Min. Illumination		Color: 0.05Lux@F1.6; B/W: 0.005Lux@F1.6
S/N Ratio		More than 50dB
Camera Features		
Day/Night		Auto(ICR) / Color / B/W
Backlight Compensation		BLC / HLC / WDR
White Balance		Auto, ATW, Indoor, Outdoor, Manual
Gain Control		Auto / Manual
Noise Reduction		Ultra DNR (2D/3D)
Privacy Masking		Up to 24 areas
Digital Zoom		16x
Lens		
Focal Length		2.7mm~11mm(4x Optical zoom)
Max Aperture		F1.6~ F2.8
Focus Control		Auto / Manual
Angle of View		H: 93.5° ~ 30.24°
Close Focus Distance		100mm~ 1000mm
PTZ		
Pan/Tilt Range		Pan: 0° ~ 355°; Tilt: 0° ~ 90°, auto flip 180°
Manual Control Speed		Pan: 0.1° ~100° /s; Tilt: 0.1° ~60° /s
Preset Speed		Pan: 100° /s; Tilt: 60° /s
Preset		300
PTZ Mode		5 Pattern, 8 Tour, Auto Pan, Auto Scan
Speed Setup		Human-oriented focal length/ speed adaptation
Power up Action		Auto restore to previous PTZ and lens status after power failure
Idle Motion		Activate Preset/ Pan/ Scan/ Tour/ Pattern if there is no command in the specified period
Video		
Compression		H.264 / MJPEG
Resolution		1080P(1920×1080)/ 720P(1280×720) / D1(704×576/ 704×480) /CIF (352×288/352×240)
Frame Rate	Main Stream	1080P/ D1 (1 ~ 25/30fps), 720P(1~50/60fps)
	Sub Stream 1	D1/CIF(1 ~ 25/30fps)
	Sub Stream 2	720P/ D1/ CIF (1 ~ 25/30fps)
Bit Rate		H.264: 448K ~ 8192Kbps, MJPEG: 5120K ~ 10240Kbps

DH-SD22204T-GN

Intelligent Function	
IVS (optional)	Tripwire, Intrusion, Abandoned Object Detection, Missing Object Detection,
Face Detection	Support
Network	
Ethernet	RJ-45 (10/100Base-T)
Protocol	IPv4/ IPv6, HTTP, HTTPS, SSL, TCP/IP, UDP, UPnP, ICMP, IGMP, SNMP, RTSP, RTP, SMTP, NTP, DHCP, DNS, PPPOE, DDNS, FTP, IP Filter, QoS, Bonjour, 802.1x
ONVIF	ONVIF Profile S
Max. User Access	20 users
Smart Phone	iPhone, iPad, Android, Windows Phone
Auxiliary Interface	
Memory Slot	Micro SD, Max 128GB
General	
Power Supply	DC12V, POE
Power Consumption	10W
Working Environment	-30°C ~ 60°C / Less than 90% RH
Ingress Protection	IP66 & IK10
Dimensions	Φ122 (mm) x 89 (mm)
Weight	0.6kg

DH-SD22204T-GN

Dimensions



Dahua Technology Co., Ltd.

1199 BinAn Road, Binjiang District, Hangzhou, China

Tel: +86-571-87688883

Fax: +86-571-87688815

Email: overseas@dahuatech.com

www.dahuasecurity.com

*Design and specifications are subject to change without notice.