Multi-Sensor Multi-Person Tracking on a Mobile Robot Platform

To the faculty of Mathematics and Computer Science
of the Technische Universität Bergakademie Freiberg
approved

THESIS

to attain the academic degree of
Doktor-Ingenieur (Dr.-Ing.)

submitted by Dipl.-Inf. (FH) Peter Poschmann
born on October 24, 1984 in Dresden

Reviewers: Prof. Dr.-Ing. Bernhard Jung, Freiberg
Prof. Dr.-Ing. habil. Hans-Joachim Böhme, Dresden
Prof. Dr. Horst-Michael Groß, Ilmenau

Date of the award: February 1, 2018

Abstract

Service robots need to be aware of persons in their vicinity in order to interact with them. People tracking enables the robot to perceive persons by fusing the information of several sensors. Most robots rely on laser range scanners and RGB cameras for this task. The thesis focuses on the detection and tracking of heads. This allows the robot to establish eye contact, which makes interactions feel more natural.

Developing a fast and reliable pose-invariant head detector is challenging. The head detector that is proposed in this thesis works well on frontal heads, but is not fully pose-invariant. This thesis further explores adaptive tracking to keep track of heads that do not face the robot. Finally, head detector and adaptive tracker are combined within a new people tracking framework and experiments show its effectiveness compared to a state-of-the-art system.

Declaration

I hereby declare that I completed this work without any improper help from a third party and without using any aids other than those cited. All ideas derived directly or indirectly from other sources are identified as such.

I did not seek the help of a professional doctorate-consultant. Only those persons identified as having done so received any financial payment from me for any work done for me.

This thesis has not previously been submitted to another examination authority in the same or similar form in Germany or abroad.

June 6, 2017 Dipl.-Inf. (FH) Peter Poschmann

Acknowledgements

This thesis would not have been possible without the help and dedication of Hans-Joachim Böhme, who provided me with the opportunity to work on exciting projects and write this thesis. He always knew how to motivate me when I was desperate and willing to give up.

I am thankful to Bernhard Jung for accepting me as an external PhD student, giving me a lot of freedom, and having advice when needed.

I wish to thank Matthias Rätsch, who had a major influence on the direction of this thesis, for pushing me when necessary.

I would also like to thank the ESF and Cognitec Systems GmbH for partially funding my research.

Contents

Chapter 1

Introduction

Autonomous robots have been a research topic for quite some time now [Schwartz et al., 1987], and while first being brought to laboratory environments and museums [Burgard et al., 1999], they start conquering streets [Levinson et al., 2011] and help training rehabilitation patients [Groß et al., 2016]. In the beginning, the research was about building environment representations [Elfes, 1987] and navigating safely [Koditschek, 1987], and has shifted towards human-robot-interaction since [Schulte et al., 1999]. Examples are service robots that act as museum tour-guides or entertain senior citizens in a retirement home. In these scenarios, the robot has to keep track of surrounding persons, so it must be able to reliably detect and track them.

Other applications of people detection and tracking are autonomous cars, driver assistance systems, and even surveillance systems. In these outdoor scenarios, the persons are further away from the cameras and usually do not pay attention to them. Datasets used for evaluation often show whole pedestrians because of their distance to the camera. This work on the other hand focuses on indoor scenarios, where people interact with a robot and are thus much closer to its sensors.

This thesis is about detecting and tracking people in the vicinity of the robot fusing the information obtained by several sensors. By knowing the position of persons, the robot can approach them and engage in interaction. But interaction is not limited to the verbal part: Establishing eye contact between robot and human is a strong cue to confirm intentions and give the human a stronger impression of being perceived, which makes the interaction feel more natural [Ito et al., 2004, Yoshikawa et al., 2006]. Therefore, the focus is on

enabling an awareness behavior by estimating the head or face position of the persons.

1.1 Projects and applications

This work began with a project named "Person detection and face analysis for human-machine interaction", supported by ESF grand number 100071902. Shortly after the project started, the task was re-focused on person detection and tracking, as this was the more pressing issue at the time. The first version of the people tracker was finished as part of this project and is described in this thesis.

Within the Artificial Intelligence and Cognitive Robotics group at the University of Applied Sciences Dresden, the people tracker was used multiple times to catch the attention of visitors of the "Lange Nacht der Wissenschaften", where the robot was standing in front of a room with other robots, trying to lure the visitors in. This was successful, as many people got curious after the robot looked at and spoke to them.

Another application is the museum tour-guide robot of project "Intelligent Interactive Mobile Service and Assistance Systems" (ESF grand 100076162), where the robot is partially controlled by a human as part of a Wizard of Oz setting [Poschmann et al., 2012a] (see figure 1.1). Navigation, people tracking, and looking at visitors are taken care of by the robot itself, which enables the human operator to focus on the spoken dialog.

Besides its successful application, the people tracker is far from perfect and has several critical issues. In later projects, solutions for these problems were developed and eventually resulted in a revised version of the tracker, being the content of the second main part of this thesis.

1.2 Robot and sensors

Two very similar kinds of robots are used for the mentioned applications: A5 and G5 of the SCITOS series by Metralabs GmbH. They have a ring of sonar sensors close to the ground, two laser range scanners above those, a head that can be turned by 360 ° and tilted for around 27 °, four small cameras just

Figure 1.1: Interaction with a museum tour-guide robot.

above the head, whose images are stitched to form a 360 ° panoramic image, and a single PMDTec CamCube time-of-flight (TOF) camera on top of a pan-tilt-unit. A display is used to show information and allows for immediate human-robot interaction. See figure 1.2 for images of the robots.

The sonar sensors are fairly unreliable and have low resolution. The laser scanners deliver similar information, but much more accurate. Therefore, the sonar sensors are not used for tracking persons. The omni-directional image, which is created using the four cameras above the head, is depicted in figure 1.3. It is of low quality, as it only has a height of 240 pixels and is fairly blurry. The advantage though is the large horizontal field of view, which enables the robot to see in every direction at all times. The laser scanners and omni-directional camera are the main sensors used for tracking. Additionally, the TOF camera is used, but the field of view is small with only about 40 ° horizontally and vertically. Therefore, this thesis does not focus on depth perception capabilities.

All computations are done on the PC inside the robot, so there is no dependency on external hardware, Wi-Fi connections, or low latency communication. There is no graphics card, so GPU-acceleration is not available. Thus, all computations have to be carried out on the single CPU inside the robot. As it is shared with other important processes like navigation, the people tracking has to be efficient.

(a) August, a Metralabs SCITOS A5. (b) Tesaro, a Metralabs SCITOS G5.

Figure 1.2: Two of the robots the people tracking is developed for.

1.3 Focus of this thesis

The persons that are to interact with the robot should have the impression that the robot perceives them. This impression is triggered by the robot looking at them. To be able to do so, the robot must know the position of the person's eyes, face, or head. The low image quality makes it difficult to identify the eyes directly without considering the surrounding context, which rules them out. The face is only visible if the person is looking in the general

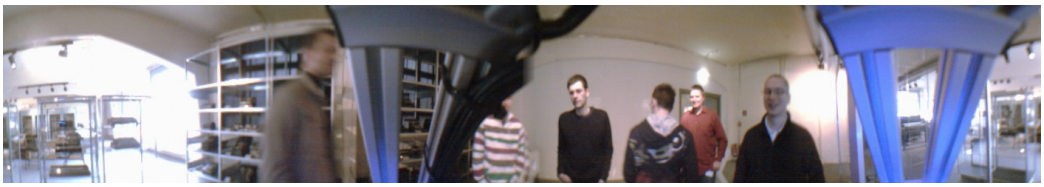


Figure 1.3: Omni-directional image, which is stitched from the images of four cameras. It features image noise, over-exposure, stitching artifacts, blurriness, motion blur, and has a resolution of 1406x240 pixels.

direction of the robot and might get lost if the person looks away. Therefore, this thesis focuses on the detection and tracking of heads, if possible.

Including context around the head, such as the shoulders or upper body in general, can increase the detection performance. However, when faced with larger groups of persons, occlusions become an issue. By focusing on the head, this is not as big of a problem, as it is the least likely body part to be occluded, especially in interaction scenarios.

This thesis explores the development of a head detector, which should be fast, reliable, and work independently from the person's viewing direction. As it is challenging to meet all three requirements, the thesis continues with the development of a head tracker that is extended by an adaptive component. By adapting to the specific head and image conditions, the tracker has the potential to surpass the detector's performance with a moderate increase in computation time. Head detector and tracker are then combined into a people tracking framework that makes use of several sensors. Experiments evaluate the effectiveness of the new approach compared to the first people tracker version.

Other aspects of this thesis are the fusion of multiple sensors and challenges that come from multi-target tracking. The former is about the integration of several sensors to increase the tracking's robustness, where e.g. the laser scanner can only sense the people's legs and thus cannot accurately estimate the head position. The latter is about tracking multiple persons that should not be confused with one another. This may happen if persons occlude each other, which typically occurs in situations where the robot interacts with groups of persons.

1.4 Outline

Chapter 2 gives an overview of the state of the art and challenges of multi-modal multi-person tracking. Based on this, a simple people tracker is introduced in chapter 3. Though successfully utilized on multiple occasions, some weaknesses are recognized and improved in the following parts of this thesis. Chapter 4 is about the detection of heads as opposed to faces and chapter 5 introduces an adaptive short-term tracker, which helps to keep track of detected targets in image sequences. Both are combined to create a multi-head tracker in chapter 6. These developments then lead back to people

tracking in chapter 7, where a revised version of multi-modal multi-person tracking is proposed based on the previous work. Chapter 8 concludes this thesis.

Chapter 2

State of the Art

Several components are necessary in a people tracking framework. In the beginning, there usually needs to be a detection, where the sensors are scanned for the presence of persons. Once a person is found, the tracker is initialized at its position. The detection is not perfect though and makes mistakes in the form of misses, where a real person is not detected, and false positives, where a person is reported that does not exist.

Tracking is the continuous estimation of a person's position over time. Tracking-by-detection fuses found person locations to form tracks. Other tracking approaches do not rely on a detector or other prior knowledge about the object to track and just search for the position that best matches the previous object appearance. Filtering fuses detections or raw sensor measurements to estimate the state of a person. In addition to the position, the state may also include orientation and velocity, and in some cases even an appearance model or other features that describe the person or its behavior.

There are two ways of handling the tracking of multiple persons: Estimating the joint state of all persons, which is a high-dimensional problem, or tracking them individually and independently of each other. The latter is usually chosen because of its computational efficiency, but in this case the detections have to be associated to the correct tracks. This problem is known as data association.

On top of assigning detections to tracks, data association has to cope with false as well as missed detections. Misses especially occur when a person is occluded, e.g. when she walks behind a pillar or stands behind another

person. Some approaches handle occlusions explicitly, while others do not. Another reason for a missed detection is the disappearance of a person, which happens when she leaves the sensor's field of view. This needs to be detected and the tracking has to be terminated, as otherwise it might have a negative influence on subsequent data associations.

Typically, a robot is equipped with several sensors. Each sensor has strengths and weaknesses. A laser scanner is able to sense most opaque obstacles and can confirm their presence, but it cannot as easily differentiate between a person's leg and an office chair, for example. A camera image on the other hand contains much more information, which enables discriminating persons from other objects. It may miss some of them though, as it depends on the appearance, which may change drastically depending on view point, pose, lighting, and other conditions. Fusing the information of several sensors has a positive impact on tracking, as they can compensate for each other's weaknesses.

2.1 Detection

Detection is about finding regions in sensor measurements that contain an object of interest. In case of a laser range scanner that is mounted below waist level, the objects of interest are usually legs if detecting persons is the task at hand. The detection algorithm then finds sets of range measurements that are caused by legs. For still images, the detector might return axis-aligned bounding boxes that indicate the position and scale of faces, heads or whole pedestrians.

2.1.1 Laser range scan

To detect persons in laser range scans, the measurements are usually divided into segments based on jump-distance clustering [Spinello et al., 2008], single-linkage clustering [Lau et al., 2010], or similar algorithms. Then those segments are classified as to whether they are caused by a person or not. Many robots, including the ones used for this thesis, have laser scanners mounted well below the waist level. In this case the task is to find measurements that are caused by the legs of persons, so usually some kind of leg detection is necessary.

A simple approach is to remove all readings that are explained by the map of the environment using background subtraction [Zhao and Shibasaki, 2005, Lau et al., 2010], and regard the remaining segments as legs. This may miss persons close to static obstacles and is prone to false positives caused by other objects. Considering only local minima [Schulz et al., 2003a] is problematic when encountering groups of persons with many partial occlusions, as the laser scan is quite cluttered in this case. Relying on motion detection [Yuan et al., 2015] misses persons that do not move, as is often the case in interaction scenarios with a non-moving robot.

More advanced approaches extract features from each segment and use these to classify them as either leg or non-leg. The classification can be based on simple hand-made rules [Xavier et al., 2005, Müller et al., 2007] or machine learning [Arras et al., 2007, Leigh et al., 2015].

When the legs are found, they must be associated to each other in a pairwise manner to yield person detections. This can be achieved using a simple nearest neighbor algorithm [Müller et al., 2007] or with more complex approaches that analyze movement patterns [Taylor and Kleeman, 2004, Zhao and Shibasaki, 2005, Arras et al., 2008].

Some approaches do not start by identifying single legs, but detect persons as a whole according to certain patterns [Bellotto and Hu, 2009, Weinrich et al., 2014].

2.1.2 Camera image

Image-based detection analyzes regions of the image and classifies them according to whether they contain the object of interest or not. The main difference between many detectors is in the choice of features and classifier. The famous Viola-Jones-Detector [Viola and Jones, 2004] uses a cascade of classifiers on Haar-like features that are efficiently computed using an integral image. It was first known for its face detection performance [Viola and Jones, 2001, Lienhart and Maydt, 2002, Huang et al., 2005], but was also adapted to other object classes like upper bodies [Kruppa et al., 2003].

The arguably most well known approaches for pedestrian detection are based on histograms of oriented gradients (HOG) and linear support vector machines [Dalal and Triggs, 2005, Pedersoli et al., 2009]. The original HOG features are highly correlated, so a variation was proposed that contained more

information while having less dimensions [Felzenszwalb et al., 2010]. This work also explored the idea of a part-based model that takes into account that the object of interest might by articulated, which is the case for pedestrians.

As opposed to the mentioned model, where the parts were implicitly defined during the training procedure, other part-based approaches for pedestrian detection explicitly define and annotate the parts [Wu and Nevatia, 2007, Andriluka et al., 2008].

There are many more features in the pedestrian detection literature besides Haar wavelets and HOG. Other examples include edgelets [Wu and Nevatia, 2007], shape context [Andriluka et al., 2008], CENTRIST [Wu et al., 2011], or integral channel features [Dollár et al., 2009]. See [Simonnet et al., 2012, Dollár et al., 2012, Benenson et al., 2014] for surveys on pedestrian detection.

In robotics, often the intrinsic and extrinsic camera calibration is known, which can be used to estimate the ground plane position within the image. It can be used in conjunction with the typical heights of humans to limit the region of interest. Due to the removal of regions that are unlikely to represent persons, the efficiency of the detector is increased [Sudowe and Leibe, 2011]. Another approach limits the regions of interest based on detections of other sensors [Blanco et al., 2003, Spinello et al., 2010], but in this case has to rely on a low miss rate of those sensors.

Several detection algorithms can be used in conjunction, e.g. motion, face, upper body, and full pedestrian detection [Volkhardt et al., 2013]. Some people trackers rely on an RGB-D camera or GPU-acceleration [Linder et al., 2015], while the work of [Yuan et al., 2015] is specialized on cameras that observe the shoes of the persons to track.

2.2 Filtering

In person tracking, filtering estimates the current state $\mathbf{x}_k$ of a person given its previous state $\mathbf{x}_{k-1}$ and current sensor measurement $\mathbf{z}_k$, where k is a discrete time index. This assumes a Markovian discrete-time process that evolves from time t_{k-1} to t_k, where the current state only depends on the immediately preceding one and measurements only depend on the current state.

The process is described by two time-dependent equations, the motion and measurement equation, which both are subject to white noise:

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \mathbf{w}_k) \tag{2.1}$$

$$\mathbf{z}_k = h_k(\mathbf{x}_k, \mathbf{v}_k) \tag{2.2}$$

Eq. 2.1 describes the dynamics of the tracked person with process noise $\mathbf{w}_k$ and eq. 2.2 computes the expected measurement with noise $\mathbf{v}_k$. Process and measurement noise are uncorrelated and independent of each other, as is past and future noise. They capture the uncertainty of the dynamic process and the inaccuracy of the measurements. Those uncertainties are taken care of by probabilistic approaches [Thrun et al., 2005], yielding probability density functions $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ for the motion and $p(\mathbf{z}_k|\mathbf{x}_k)$ for the measurement.

The Bayes filter recursively estimates the state probability density function $p(\mathbf{x}_k|\mathbf{z}_{1:k})$, where $\mathbf{z}_{1:k} = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$ combines all measurements from time t_1 through to time t_k:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} \tag{2.3}$$

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1} \tag{2.4}$$

$$p(\mathbf{z}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})d\mathbf{x}_k \tag{2.5}$$

The integrals in eq. 2.4 and 2.5 cannot be computed in closed form for arbitrary probability density functions. If process and measurement noise are Gaussian and the motion and measurement equations are linear, then the Kalman filter delivers a closed-form solution to the recursive state estimation problem [Kalman, 1960].

If the equations are non-linear, then they may be linearized by a first-order or second-order approximation, yielding the extended (EKF) or unscented Kalman filter (UKF) [Julier and Uhlmann, 1997a], respectively. In case

of non-linear equations and/or non-Gaussian noise, the particle filter (PF) can be used to approximate the probability density function using samples [Gordon et al., 1993, Doucet and Johansen, 2009].

Many people tracking approaches circumvent non-linear equations by choosing linear motion models and transforming the detections into a common coordinate system before updating the filter, which enables the usage of the vanilla Kalman filter [Spinello et al., 2010, Volkhardt et al., 2013, Linder et al., 2015]. Most trackers use a near constant velocity model or random walk (Brownian motion), whereas [Spinello et al., 2010] use both and instantiate two Kalman filters per person, choosing the one whose prediction is closest to the measurement.

[Bellotto and Hu, 2010] investigated the performance differences between the EKF, UKF and PF when using non-linear motion and measurement models. The EKF performed worse across the board, while the PF used the most resources with a particle count of either 500 or 1000.

2.3 Data association and track management

In multi-target tracking, there are several tracks and detections. Each person is assumed to cause either one detection (hit) or none (miss) and a detection is caused by the presence of a person (true positive) or clutter (false positive). Data association is concerned with finding the correspondences between tracked persons and detections, so the filtering algorithm uses the correct data to update the states of the persons.

The number of persons is unknown beforehand, so a track has to be initiated whenever a new person appears in the sensor's field of view. In a similar fashion, a track has to be terminated when the corresponding person leaves the field of view. This is made difficult by false and missed detections. A false detection may lead to a track that does not correspond to a real person, whereas a missed detection may lead to the removal of a track that is still needed. False detections and misses have a negative influence on data association, as have wrong and missing tracks.

The simplest approach is a nearest neighbor data association that assigns tracks and observations based on their distances [Montemerlo et al., 2002]. Unassigned detections are used to initiate new tracks. Once a track and

observation were chosen, they cannot be assigned again, so each track is associated to at most one observation and vice versa. In contrast to that, the joint-probabilistic data association [Fortmann et al., 1983] weights the influence of each observation based on its probability to be caused by the target, effectively assigning fractions of all observations to each track.

False positives can be reduced by delaying track initiation [Linder et al., 2015] or introducing a probability of existence to each state vector [Volkhardt et al., 2013]. This comes at the cost of an increased miss rate, though. [Munz et al., 2010] extend the existence probability to three cases instead of two: no object, any non-relevant object, and relevant object. This allows for a better modeling of heterogeneous sensor capabilities.

More complicated data association algorithms do not resolve ambiguities immediately, but instead keep multiple association hypotheses, postponing the decision until more information is available. The track-splitting filter [Smith and Buechler, 1975] keeps hypotheses for each track, which may lead to observations being assigned to more than one track. The multiple-hypothesis filter [Reid, 1979, Cox and Hingorani, 1996] on the other hand keeps hypotheses over all tracks and also handles the initiation and termination of tracks. Both approaches prune their hypothesis trees to stay computationally tractable. Nevertheless, they have exponential complexity and thus are less efficient than their simpler counterparts. A similar approach is used by [Schulz et al., 2003b], where a Rao-Blackwellized particle filter manages the hypotheses.

For multi-sensor person tracking, a simple approach to data association usually suffices, as the sampling rate of the sensors is high (usually several times a second) and the detector performance is far more important, as high false positive rates are the biggest problem. The tracking approaches perform quite similar when combined with detectors that have a low false positive rate [Linder et al., 2016]. See [Cox, 1993] for a general survey on data association algorithms.

Data association errors can be reduced with more sophisticated motion models that lead to better movement predictions [Luber et al., 2010]. Additionally, some ambiguities can be resolved by including appearance information into the distance function, e.g. in the form of histogram comparison [Yuan et al., 2015] or scores of target-specific classifiers that are trained on-line [Song et al., 2010, Breitenstein et al., 2011].

2.4 Sensor fusion

Robots typically have multiple sensors of different kinds. Their data is complementary and can be combined to improve the tracking performance. Multi sensor data fusion is the integration of information from several sensors to acquire more robust and accurate knowledge than each of the sensors alone could provide. This may result in a larger field of view, more accurate estimations, or less misses.

Filtering (sec. 2.2) fuses data of a single sensor over time. By combining the measurements of several sensors into a single measurement vector and respecting their correlations, it can be used to integrate the data of multiple sensors, but needs them to measure simultaneously.

If the measurement errors of the sensors do not correlate with each other, they can be used to update the estimated state independently of each other, with each sensor having its own measurement model. In this case, the measurements are not required to come in at the same time and the sensors can have different update rates. This approach is used by many robot-based person tracking frameworks [Bellotto and Hu, 2010, Volkhardt et al., 2013, Yuan et al., 2015]. Nevertheless, measurement models may be identical for different sensors if the observations are transformed into a common coordinate system.

In case of unknown correlations between the measurements, covariance intersection is an alternative to the measurement update of a Kalman filter [Julier and Uhlmann, 1997b]. It prevents an overly optimistic state estimation in case of correlated data, but needs the measurements or detections to be in the same coordinate system.

Detections can also be fused before filtering. By aggregating them over a short time window, detections of the same person by different sensor cues can be combined before integrating them into the filtering algorithm [Linder et al., 2015].

Another technique is track-to-track fusion, where each sensor operates its own filtering algorithm and reports the filtered tracks, which are fused afterwards. In this case, the tracks are correlated and the fusion solution might be suboptimal depending on update rate, feedback to the sensor filters, and memory of the fusion algorithm [Tian and Bar-Shalom, 2009].

Bayes filters (and tracking approaches in general) assume measurements to arrive in sequence, so the timestamp associated with a measurement is never

older than the timestamp of previously processed measurements. When using multiple sensors, this assumption does not necessarily hold anymore, and the system has to cope with out-of-sequence measurements. This problem is amplified by differing detection times depending on the sensor, as detecting humans in camera images usually takes much more time than detecting humans in laser scans.

Discarding the out-of-sequence measurements might lead to a significant loss of data, as certain sensors usually lag behind others and thus may never be incorporated into the state estimation. Simple solutions are buffering and reprocessing. Buffering stores measurements for some time and re-orders them before processing, which leads to a delay and decreases the reaction time of the system. Reprocessing stores measurements and estimated states. On arrival of a delayed measurement, the system is rolled back to the associated timestamp and measurements including the new one are reprocessed, leading to increased memory requirements and processing time.

There are solutions with less memory and processing requirements compared to reprocessing, e.g. for Kalman filter [Zhang et al., 2005, Bar-Shalom et al., 2004] and particle filter [Orguner and Gustafsson, 2008], but they are intended for single target tracking or assume the data association to remain unchanged when adding the delayed information. Approaches that tackle the multi-target case, e.g. building upon multiple-hypotheses tracking, [Mallick et al., 2002], are much more sophisticated.

Out-of-sequence measurements are rarely mentioned in the multi-sensor people tracking literature. An exception is [Volkhardt et al., 2013], that integrate a delayed detection by predicting its position in the current time according to the motion model of the associated track.

Further challenges of multi-sensor data fusion and possible solutions are discussed in [Khaleghi et al., 2013].

2.5 Conclusion

Some basic components are often used by robot-based people trackers. Laser range scanners and cameras are mostly utilized. Persons are detected within the sensor's measurements. Those detections may or may not be transformed into a common coordinate system. A simple data association finds the right track for each detection. Finally, a filtering algorithm is used to update the tracks. With these components, a simple people tracker can be build.

Chapter 3

People Tracking - Version 1

This chapter describes a simple approach to people tracking, which is based upon the state of the art and adopts many of its ideas. It is used to identify the biggest weaknesses and problems, which are then worked on in the following chapters to lead to a more reliable people tracking framework. This first tracker version is based on a tracking-by-detection approach, where persons are detected by several sensors and those detections are associated to tracks.

To simplify the sensor fusion, the observations are transformed into the three-dimensional world coordinate system defined by the map. The map is learned in advance by a SLAM approach and represents the environmental model the robot is operating in.

The position estimation of a person is not equally accurate across the sensor cues: The laser scan does not contain any information about a person's height, whereas the detection of a person in an image only gives rough information about its distance to the camera. These uncertainties are explicitly modeled using Gaussians. In this case, the continuous estimation of the person's positions can be carried out using a Kalman filter. The people tracker described here is published in [Poschmann et al., 2012b].

Figure 3.1 illustrates the steps of the people tracking algorithm. The sensor cues receive measurements, detect persons, and transform those detections to the map frame. The tracker is responsible for data association, filtering, track removal and initiation, and output generation based on the confirmed tracks.

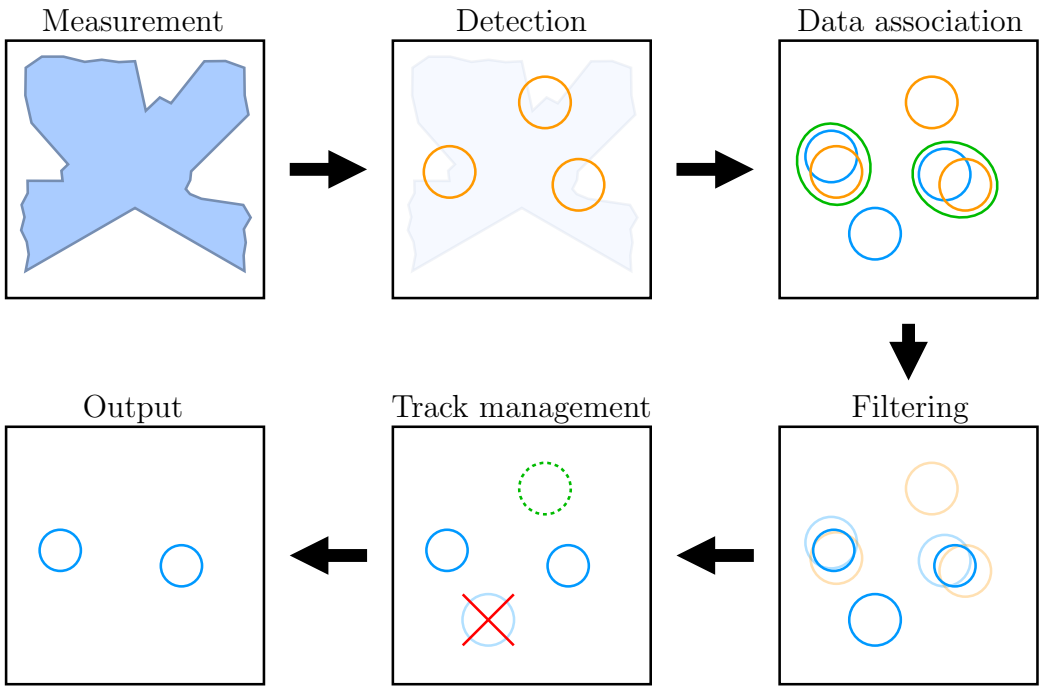


Figure 3.1: Steps of the people tracking algorithm. Orange circles represent detections, while tracks are blue (existing) or dashed green (newly initiated).

3.1 Detection

The sensor measurements are analyzed for the presence of persons. The resulting detections are transformed into three-dimensional Gaussian estimates of the head positions in the world coordinate system, based on the robot pose. Thus, the tracker does not need to be changed when a new sensor cue is added, as all observations are handled in the same way.

3.1.1 Laser range scan

The detection is inspired by a group tracking and size estimation algorithm [Lau et al., 2010] and combines several clustering algorithms. First, background subtraction removes measurements that are likely to be caused by static obstacles. This step needs an occupancy grid map of the environment and a sufficiently accurate robot pose. The grid map is dilated by 20 cm to compensate for uncertain localization. Each laser range measurement that is located inside one of the occupied grid cells is removed. Only measurements that are not explained by the known static obstacles remain.

To remove noisy points, the remaining measurements are grouped using single-linkage clustering with a distance threshold of 5 *cm*, so any two points closer than that will end up in the same cluster. Then, clusters with just a single point are discarded. All remaining measurements are assumed to be caused by human legs.

Single-linkage clustering is a hierarchical algorithm. At the beginning, each point is its own cluster. Subsequently, the two closest clusters will be merged into one, until the closest distance is above a pre-defined threshold. In case of single-linkage clustering, the distance between two clusters is chosen to be the smallest possible distance, that is the distance between two points of different clusters that are closest to each other. After using this algorithm, the clusters have a minimum distance equal to the chosen threshold between each other. A similar approach is complete-linkage clustering, which uses the distance between the farthest points of two clusters. This leads to clusters whose size is at most the pre-defined threshold.

After pre-processing, single-linkage clustering is used on the remaining points with a distance threshold of 60 *cm*, which should be larger than a typical step width of a human, assuming the laser range scanner is mounted in approximately knee-height and persons are standing or slowly walking rather than running around. This will divide the measurements into groups of people, where in the best case there is only one person per group.

The next step is to estimate the leg positions within each group. Unlike [Lau et al., 2010], a complete-linkage clustering is used with a distance threshold that is in between the diameter of a single leg and the combined diameter of two legs, e.g. 25 *cm*. This leads to one cluster for each non-occluded leg, whose position is determined by the average of the cluster's points. Then, the legs are assigned to each other in a pairwise manner using a greedy nearest neighbor algorithm with a maximum distance of 70 *cm* between two legs. Remaining single legs and the center position between two assigned legs are the estimated positions of the persons. Clustering and resulting person detections are illustrated in figure 3.2.

To arrive at the final three-dimensional observations necessary for the tracker, a height of 1.5 *m* is assumed. The variance for the height is chosen very high, so its influence on the height estimation is negligible. The positional variance is chosen depending on the number of persons per group - if there is more than one, then the leg assignment might have gone wrong, and higher values are chosen for the variances.

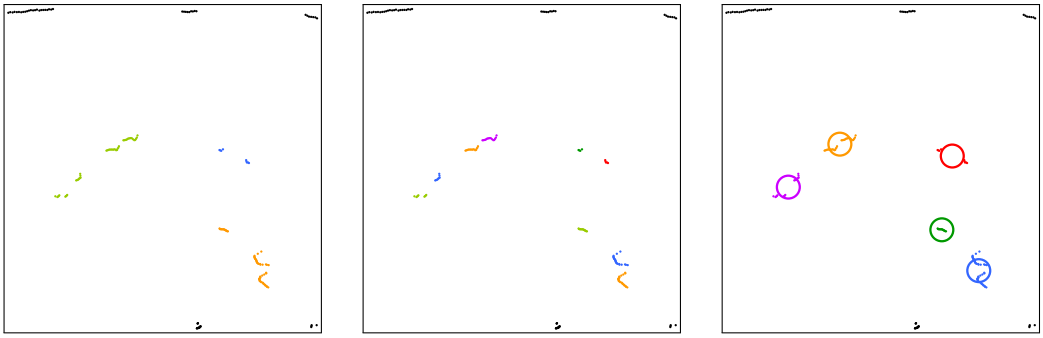


Figure 3.2: Person detection in a laser range scan using a combination of clustering algorithms. Black points represent laser range measurements that are caused by static obstacles or are regarded noise and are thus not considered to be persons. *Left:* Background measurements are removed, the remaining points are clustered into groups of persons. *Center:* The groups are further clustered into legs. *Right:* Pairwise assignment of legs leads to estimated positions of persons, as do the remaining single legs.

This clustering approach is simple and delivers reliable results with groups of persons and partial occlusions, but has problems with wide skirts, coats or bags, which leads to an over-estimation of the number of legs.

3.1.2 Camera image

The tracker should be able to operate in interaction scenarios, were persons are not too far away from the robot. In this case, the feet and legs are (partially) outside the camera's field of view, which renders full-body person detectors impractical. Therefore, the image-based detection focuses on the upper body and face of the persons. They are located using OpenCV's Viola-Jones-Detector, which is already trained and ready-to-use.

To make the detection more efficient, the region of interest inside the omni-directional image is restricted to areas where persons are expected according to the tracker. The drawback is a dependency on the detection capabilities of other sensors, mainly the laser range scanners. The face and upper body detector operate on the same image, but result in two different sensor cues. Figure 3.3 shows an example image with detections.

For reasons of simplicity, the same conventional detection approach is used on the TOF camera images. Furthermore its field of view is very limited and thus

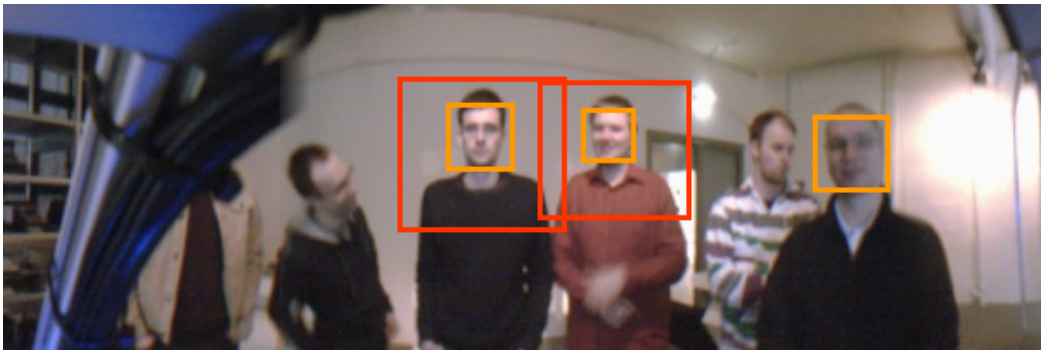


Figure 3.3: Face (orange) and upper body (red) detections.

only has little influence on the whole people tracking. Its low resolution makes using the detection algorithm fast compared to the omni-image. It has its own infrared light source, which makes it independent from the environment's lighting conditions and improves the detection rate.

The estimated eye-level within the detections is used to derive the head position. This helps the robot to establish eye-contact, as it just has to look at the estimated position. The distance from the camera is computed using the scale of the detected bounding box and a rough estimation of face and upper body size in the real world. As the distance estimation is not too accurate, the uncertainty is chosen higher than for the lateral direction. For the TOF camera, there is no need to estimate the distance like that, as it can be determined from range measurements inside the bounding box. Consequently, the distance variance is chosen lower than for the omni-directional image.

3.2 Tracking

Tracking fuses several detections to achieve a more accurate estimation of a person's state. Additionally, the association of detections across time enables the robot to continuously know which of the persons around it is its interaction partner, e.g. the person it is looking at. Because of linear motion and measurement equations, a vanilla Kalman filter is used for the state estimation of each person. Out-of-sequence measurements are integrated by reprocessing.

3.2.1 System and state

The state of a person needs to include the 3D position of the head, so the robot knows which direction to look in. This is equal to the information that is delivered by the sensor cues. Estimating the velocity improves the prediction of future positions, which is helpful for the data association and may additionally be used for navigation purposes, where the robot must prevent collisions with persons or invading their personal space. As humans rarely change their height (e.g. by crouching down) and the world of the robot is assumed to have an even ground, the velocity is only two-dimensional [Bellotto and Hu, 2006]. The state vector $\mathbf{x}_k = (x_k, y_k, z_k, \dot{x}_k, \dot{y}_k)^T$ therefore consists of five values, the 3D head position and a 2D velocity vector in the ground plane.

Each tracked person's movement is described by a near constant velocity model, which assumes the motion direction and speed to only change smoothly over time. The person's height, however, is modeled by a random walk model, as it has no associated velocity value. The motion equation (see eq. 2.1) is thus:

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + \mathbf{w}_k \tag{3.1}$$

$$F_k = F(\delta_k) = \begin{pmatrix} 1 & 0 & 0 & \delta_k & 0 \\ 0 & 1 & 0 & 0 & \delta_k \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

The tracking cycle length is given by $\delta_k = t_k - t_{k-1}$. The process noise $\mathbf{w}_k$ is caused by changes in velocity and head height and is modeled by a zero-mean Gaussian with covariance matrix Q_k:

$$Q_k = \int_0^{\delta_k} F(\tau) Q F(\tau)^T d\tau = \begin{pmatrix} \frac{1}{3}\delta_k^3 q_v & 0 & 0 & \frac{1}{2}\delta_k^2 q_v & 0 \\ 0 & \frac{1}{3}\delta_k^3 q_v & 0 & 0 & \frac{1}{2}\delta_k^2 q_v \\ 0 & 0 & \delta_k q_z & 0 & 0 \\ \frac{1}{2}\delta_k^2 q_v & 0 & 0 & \delta_k q_v & 0 \\ 0 & \frac{1}{2}\delta_k^2 q_v & 0 & 0 & \delta_k q_v \end{pmatrix} \tag{3.3}$$

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & q_z & 0 & 0 \\ 0 & 0 & 0 & q_v & 0 \\ 0 & 0 & 0 & 0 & q_v \end{pmatrix} \tag{3.4}$$

The scalars q_z and q_v are the variances per second of the height and velocity, respectively.

As the sensor cues already transform the observations into world coordinates, the measurement equation (see eq. 2.2) is the same for every sensor:

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{v}_k \tag{3.5}$$

$$H_k = H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.6}$$

The measurement noise $\mathbf{v}_k$ is modeled by a zero-mean Gaussian distribution with covariance matrix R_k, which is provided by the sensor cue together with the actual measurement.

3.2.2 Data association and track management

Observations are assigned to tracks using a greedy nearest-neighbor data association based on the Mahalanobis distance. The innovation covariance $S_k = H_k \bar{P}_k H_k^T + R_k$ is used to compute the distance, where $\bar{P}_k$ is the covariance matrix of the state after prediction. A gating procedure determines whether an association is allowed based on a distance threshold. The squared Mahalanobis distance is chi-squared distributed and the threshold is chosen to prohibit the assignment if the probability is below 1 %. In case of three dimensional observations, the threshold of the squared Mahalanobis distance is thus 11.345. Unassigned observations are used to initiate new tracks.

Each track has a confidence c that it corresponds to a real human. This confidence is used to cope with false positive detections as well as to determine whether to terminate a track that was not assigned an observation to for

some time. If the confidence exceeds a threshold λ_h, it is considered to be a real person, if it drops to zero, it is terminated.

The confidence value is based on the number of confirmations over the last second. Given the assigned observations over the last second O, the track confidence c is computed using the detection reliability r_o and the update rate f_o of the sensor cue that generated the observation o:

$$c = \sum_{o \in O} \frac{r_o}{f_o} \tag{3.7}$$

Because of the higher number of false positive detections, the laser range scanner cue is considered less reliable than the camera-based cues. The former is assigned a reliability value of 0.1, while the latter have a value of 0.8.

The resulting confidence value cannot be higher than the sensor reliability value if that sensor is the only one confirming the track. If tracks should be confirmed by at least two cues before considered to be a real person, then the threshold λ_h should be chosen slightly higher than the individual reliabilities, e.g. around 0.85. If just the visual cue should be enough, but not the laser-based ones, then the threshold must be in between their associated reliabilities, e.g. 0.5.

To prevent erratic behavior, the confidence value must fall below a second, lower, threshold λ_n before not being considered a real person anymore. This threshold can be zero if tracks are rarely confused with one another or it can be chosen to be higher than the laser-based cues, so a visual confirmation is always required, e.g. around 0.2. If a track stays without observation for one second, its confidence value drops to zero and it is terminated.

3.3 Conclusions

The proposed tracker works well in most interaction scenarios, where the task is to find at least one person for the robot to look at. In these scenarios, the persons usually stand close to the robot and face it, which makes detecting and tracking them easy. Because the robot can only look at one person at a time, the tracker does not need to find all people at all times. For a natural awareness behavior it suffices if the tracker finds more than one person over

the duration of the interaction, so the robot can change the person it looks at from time to time.

Nevertheless, the tracker reveals some weaknesses that need to be addressed. For one, it must be able to reliably detect and track persons in scenarios other than interaction, as the robot needs to make persons aware of it first. Additionally, certain interactions are not as static, for example if the robot explains a museum exhibit and the persons start moving over to it and face it. This section explains the main issues of the tracker.

3.3.1 Detection using laser range scan

The detection approach using the laser range finder assumes every non-static measurement to be caused by a human leg. This quickly leads to an overestimation of legs in case of other obstacles or spurious measurements that survive the filtering step and thus leads to false positives. On the other hand, if a person is further away or wears pants with bad reflectance properties, there might be too few measurements per leg that subsequently are removed due to the filtering and hence lead to misses.

The pairwise assignment of legs is fairly simple and might result in wrong position estimates. This is compensated for by selecting a higher positional variance, but will still influence the estimated head position. Using a Gaussian to represent the uncertainty leads to a similar problem, as the center point between the legs has a higher probability of representing the head position than any other point, e.g. above one of the legs. But in reality, the probability for the head being above one leg might not be less than for it being in between them, but this cannot be modeled by a single Gaussian distribution.

3.3.2 Detection using camera image

A person has to face the robot's camera in order to be detected. The face detector needs a frontal view, while the upper body detector is able to perceive the body from behind, but not from a side view. See figure 3.4 for examples of missed persons. This is not problematic if at least some persons in a group are looking at the robot, but there are also cases where persons are turning away from it. Imagine the robot explaining an exhibit and the museum visitors turning their attention to said exhibit. It would be bad if the robot then loses track of these visitors and assumes them to be gone.

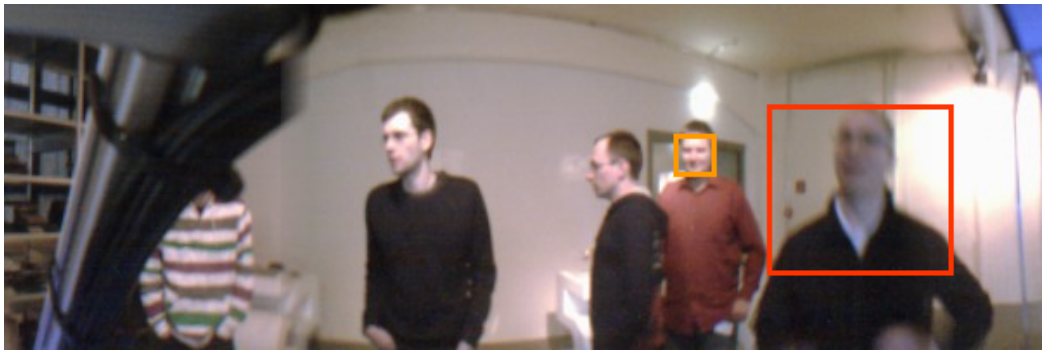


Figure 3.4: Missing face and upper body detections due to persons not facing the robot, being occluded, or being blurry because of motion.

3.3.3 Tracking-by-detection

The proposed tracker first detects persons in the sensor measurements and then associates them with existing tracks. This tracking-by-detection approach is fairly simple, but ignores prior knowledge that already exists within the tracker. That becomes most apparent when looking at the laser range scan cue, that for each laser scan tries to find persons. Associating legs to each other can result in wrong position estimates if the distance between persons is small. If the sensor cue would know the predicted positions of the persons beforehand, then this knowledge could be used to make better associations between legs.

3.3.4 Data association

The nearest neighbor data association does work well if most humans are detected, but with many false positives and misses the probability of mistakes increases. This is especially true for using the Mahalanobis distance, as the increasing uncertainty of an unconfirmed track leads to a smaller distance and thus an association with a detection further away gets more likely. While this is the intended behavior, it also increases the likelihood for wrong associations in case of missing detections.

3.3.5 Solution

All of the mentioned weaknesses are related to the sub-par performance of the detectors, especially when it comes to detect persons that do not face the robot. Developing a reliable pose-invariant detector would address most of these issues and thus have the biggest impact on tracking performance.

Chapter 4

Head Detection

The people tracker of chapter 3 uses face and upper body detection to find persons in the camera images. Both of these detectors depend on the person's pose: the face detector needs a frontal view of the head, and the upper body detector needs a view of the front or back of a person. As soon as that person turns to the side, it cannot be detected anymore, which affects the performance of the tracker. This chapter is about pose-invariant head detection, which would solve this problem.

Detecting the head instead of the face is about finding the whole object as opposed to just the frontal surface. The latter is just barely visible in profile views, which is a disadvantage compared to detecting the whole head. Including the shoulders or upper body might increase the potential for correct detection, especially for persons that are far away and appear small in the image, but is more affected by occlusions that typically occur when encountering groups. Because of this, the focus is on head detection rather than face, upper body, or pedestrian detection.

4.1 State of the art

The first step of detection is to select regions of interest. Each of these regions is a candidate for a detection and is evaluated regarding the presence or absence of a head in the second step. This step is typically divided into two parts: a feature extraction that computes a vector representing the region content and a classification of that vector into either the positive (region

represents a head) or negative class. In a third step, overlapping positive regions are reduced to a single result using non-maximum suppression, so each head is only reported once.

4.1.1 Sliding window

Most often in image-based detection, candidate regions are sampled densely across the whole image [Rowley et al., 1998, Papageorgiou and Poggio, 2000]. This sliding-window approach scans the image using a rectangular window at different positions and scales, resulting in many regions of interest, usually apart from one another by only a few pixels.

Scanning across different scales is often realized by keeping the window size fixed and downscaling the image instead, thereby creating an image pyramid. It is also possible to scale window and features instead [Viola and Jones, 2001] or to use a hybrid approach of a sparsely sampled image pyramid and feature/classifier scaling in between pyramid layers [Dollár et al., 2010]. The main drawback of the sliding window approach is the large number of candidate windows that need to be evaluated.

Alternatives are branch-and-bound methods [Blaschko and Lampert, 2008, Lampert et al., 2009], which are more tailored towards the detection of single objects instead of several, or key-point-based methods [Leibe et al., 2005], which require good image quality and generally fail in low and medium resolution [Dollár et al., 2012]. With prior knowledge of the camera geometry, robot, and environment, a ground plane estimation can help to limit the number of candidate windows [Sudowe and Leibe, 2011].

4.1.2 Classification

Many head detectors and trackers use the contour as main feature [Isard and Blake, 1998, Liu et al., 2010, Marín-Jiménez et al., 2014], sometimes accompanied by color histograms [Birchfield, 1998, Bouaynaya and Schonfeld, 2005]. This is similar to pedestrian detectors that utilize histograms of oriented gradients (HOG) [Dalal and Triggs, 2005, Felzenszwalb et al., 2010], as the contour is the most important information captured by those features in this case [Wu et al., 2011].

In the last decade, person detection was dominated by two different classification approaches: Boosting [Viola and Jones, 2004, Babenko et al., 2008, Dollár et al., 2014], which combines several weak classifiers into stronger ones, and support vector machines (SVMs) [Papageorgiou and Poggio, 2000, Dalal and Triggs, 2005, Felzenszwalb et al., 2010], that are trained by maximizing the margin of a linear decision boundary. For this particular task, kernel SVMs do not perform better than linear ones, while boosting is on par [Benenson et al., 2014].

The popular combination of HOG features and a linear SVM was also successfully applied to head detection [Benfold and Reid, 2011]. Details on feature parameters and implementation are missing though and the software was not made available.

4.1.3 Non-maximum suppression

Typically, the sliding window approach finds several overlapping positive candidates for each real head in the image. Non-maximum suppression finds the best response for each head and suppresses everything else. Mean shift can be used to find the maximal response of each detection [Dalal et al., 2006], but a simple pairwise maximum of overlapping candidates suffices [Felzenszwalb et al., 2010, Dollár et al., 2009].

In this case, the procedure goes as follows: from the highest score to the lowest, the candidates are promoted to final detections. After each promotion, every remaining candidate that overlaps sufficiently is suppressed. To determine the overlap ratio, the intersection over union of two bounding boxes is computed, also known as the Pascal criterion [Everingham et al., 2010].

4.2 Approach

The proposed head detector is based on a sliding-window approach, which seems the most promising regarding the low-resolution images and requirement to find multiple heads. HOG features can capture the head contour and potentially gradients within the face, e.g. caused by eyes, mouth, or nose, which may improve detection. The detector uses the extended HOG features by [Felzenszwalb et al., 2010] (FHOG) in conjunction with a linear support vector machine (SVM). The non-maximum suppression removes detections if their overlap ratio with another higher scoring detection exceeds 30 %.

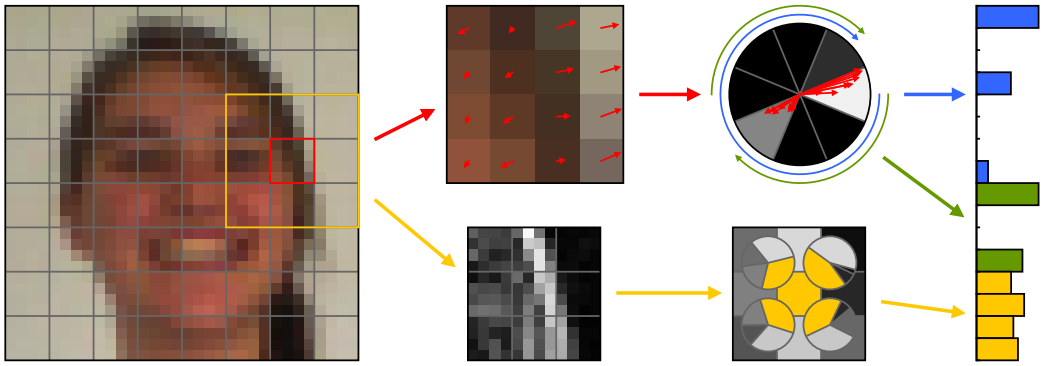


Figure 4.1: Simplified schematic representation of the FHOG cell descriptor.

4.2.1 Features

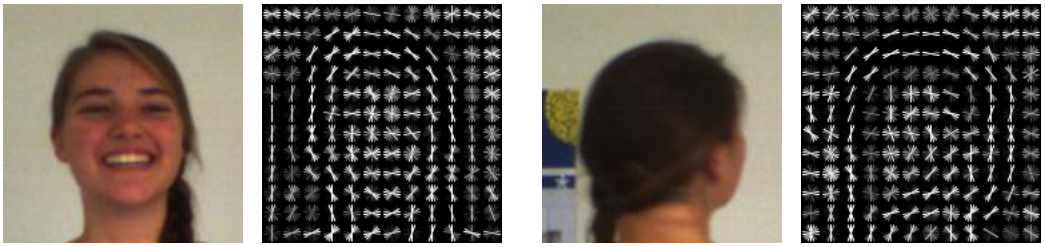
FHOG features divide the window into cells, where each cell is represented by a descriptor. Figure 4.1 shows a schematic representation of the red cell's descriptor. Within the cell, gradients are computed for each pixel. They are used to build a gradient orientation histogram, which is normalized. Normalization makes the descriptor invariant against illumination changes.

The main part of the descriptor is made up of two histograms: One of them preserves the direction of the gradients (blue in figure 4.1), while the other adds up opposite bins of the histogram (green in figure 4.1). The last part of the cell descriptor represents the gradient magnitudes compared to the neighboring cells (yellow in figure 4.1).

Figure 4.2 visualizes the unsigned orientation histograms of the cell descriptors for two images of a head. Despite the different view points, the features look similar, which is caused by the head's contour and clear distinction from the background. This suggests that the features are slightly invariant against pose changes, which allows to use a simple classifier.

The feature vector of the whole window is obtained by concatenating the descriptors of all contained cells. The size of the cells determines the stride of the sliding window. Most of the cells are then shared between neighboring window positions, so the descriptors can be computed for the whole (scaled) image before applying the sliding-window-technique instead of for each window separately.

Different variants of FHOG features are used throughout this chapter. The main variants are:



(a) Frontal head. (b) Back head.

Figure 4.2: Visualization of FHOG's unsigned orientation histograms.

FHOG: The FHOG cell descriptor contains two normalized gradient histograms, one of them unsigned (opposite orientations belong to the same bin) and the other signed (opposite orientations go into different bins), and four gradient energy values. Usually, nine unsigned orientation bins are used and histogram values are capped at 0.2 after L2 normalization.

UFHOG: Variation of **FHOG** without the signed gradient histogram, leaving only the unsigned histogram and gradient energy features.

SFHOG: Variation of **FHOG** without the unsigned gradient histogram, leaving only the signed histogram and gradient energy features.

4.2.2 Classifier

Given a feature vector $\mathbf{x}$, a linear SVM computes a score $f(\mathbf{x})$ representing the distance to its decision hyperplane

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \rho, \tag{4.1}$$

where $\mathbf{w}$ is the weight vector and ρ is the bias. The sign $y = \text{sgn}(f(\mathbf{x}))$ of the score indicates the class, which can be positive or negative.

A schematic representation of linear decision boundary and training examples can be seen in figure 4.3. Squares represent positives, while circles stand for negatives. The darker area around the decision boundary is the margin, which the training procedure tries to maximize in order to achieve the best separation between the classes. Training examples on the margin's edge are defined to have a distance of 1 to the boundary.

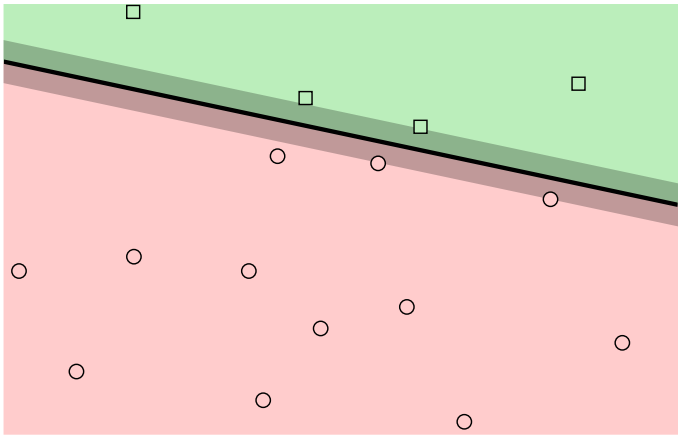


Figure 4.3: Schematic representation of training examples and resulting decision boundary of a support vector machine.

New feature vectors are classified as either positive or negative depending on which side of the decision boundary they are located. Vectors on the green side (figure 4.3) have distances greater than zero, while vectors on the red have distances lower than zero. To influence the sensitivity, thresholds other than zero can be chosen, which results in decision boundaries parallel to the default boundary.

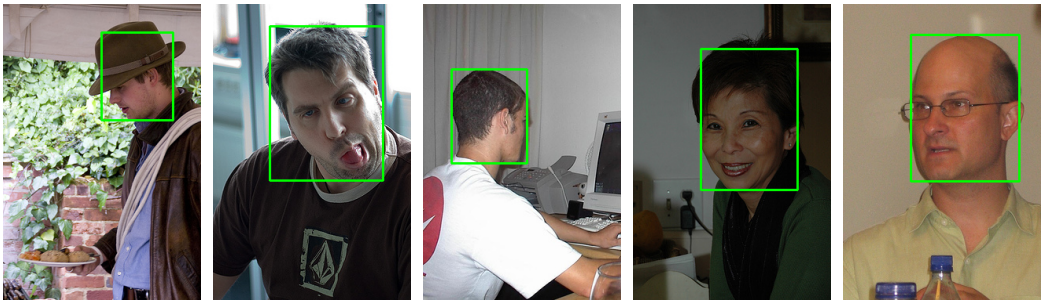
To compute the SVM score (hyperplane distance) of every possible window position of an image, the SVM's weight vector is convolved over the feature images of the pyramid and the bias is added on top. This makes it more efficient than using a kernel SVM, which would need several convolutions, one for each support vector.

If probabilistic output is necessary, then it can be computed by transforming the score using a sigmoid function [Platt, 1999, Lin et al., 2007]:

$$p(y = 1|\mathbf{x}) = p(\mathbf{x}) = \frac{1}{1 + \exp(af(\mathbf{x}) + b)} \tag{4.2}$$

4.3 Dataset

The dataset used for training and evaluating the detector consists of images selected from the Pascal Visual Object Classes (VOC) Challenge [Everingham et al., 2010]. There already exist images with annotated heads, but those



(a) Occluded (b) All (c) Upright (d) Frontal (e) Profile

Figure 4.4: Examples of head annotation categories.

annotations only show the visible part and not the (potentially occluded) total extent of the head. Additionally, they are inconsistent, sometimes leaving space between head and bounding box border, while at other times focusing on the face rather than the whole head. Therefore, new head annotations were specifically created for the proposed head detector.

The heads are marked by a rectangular, axis-aligned bounding box that closely encapsulates the skull, ignoring hairstyle, hats, glasses, and other accessories. In some cases, the bounding box could only be guessed because of that. Figure 4.4 shows examples of annotated bounding boxes. There are 283 images within this dataset, with heights ranging from 260 to 500 pixels and widths between 191 and 500 pixels.

There are two types of annotations: considered heads and ignored heads. The latter category is for heads that are artificial (e.g. paintings), occluded, or otherwise do not fit into the positive training sample class. Heads in this category are not included in the training set and detections that match these annotations in the evaluation do not count, neither as true positive, nor as false positive.

Several head classes are annotated and evaluated regarding classification performance, from all heads down to only upright frontal heads. In all of these cases, the same heads are annotated, but the distinction between considered and ignored is changed. A head might be considered in the all heads category, but ignored in the upright frontal category. This is because the detector should not be punished for finding heads of other categories, even if it was not trained on them, as in the end the task is to distinguish heads from anything else. The idea behind the categorization is better performance, as finding heads in all kinds of orientations might be a harder task and lead to more false positives than finding heads of persons looking towards the robot.

All has all heads annotated that are not too small (above around 20 pixels), but certain heads are ignored: those that are only partially within the image (on the image border), those that are out-of-focus, those that are smaller than 40 pixels, and those that are not real, e.g. in paintings or television, and those that have more non-occluded than occluded parts. Examples for occlusions are headdresses like hats (figure 4.4a) or headscarfs, dark sunglasses, or cups that are being held in front of the head. Overall there are 601 annotations with 399 being considered and 202 ignored heads. The average height of the considered bounding boxes is 104 pixels.

Upright additionally only considers heads that are upright, meaning that they are not seen from below or above. A roll and/or pitch angle of around 15 ° is allowed, everything above that is ignored. See figure 4.4b for a head that is not considered upright. The decision was made by imagining a straight line from the bottom of the head to the top and measuring its angle from the y-axis. For heads viewed from the front, the position of the eyes inside the bounding box was used as an indicator, they should not be considerably below the half or towards the top. 310 considered heads remain in this class.

Frontal contains upright heads that have a yaw angle below 45 °. Both eyes must be visible and not be occluded by the nose. Both eyes must not be completely within one side of the bounding box, so at least part of an eye must be on the other half, see figure 4.4d for a borderline example of a frontal head. Contains 158 heads to be considered, which accounts for 39.6 % of all heads.

Profile contains upright heads that have a yaw angle between 45 ° and 90 °. Both eyes must be completely on one side of the bounding box and at least one eye and the nose must be visible. Figure 4.4e shows a profile view, while figure 4.4c shows a head that is not considered profile anymore, as the person looks away from the camera. Contains 125 considered head annotations, which accounts for 31.3 % of all heads.

Frontal+Profile contains the considered heads of Frontal and Profile, which are upright heads with a yaw angle of at most 90 °. Contains 283 considered head annotations, which accounts for 70.9 % of all heads.

Other contains all the heads that are not considered Frontal or Profile, which are heads tilted to the side or seen from the back, above or below. Figures 4.4b and 4.4c are examples of annotations that belong into this category. Contains 116 considered heads, which accounts for 29.1 % of all heads.

4.4 Training

The training dataset is a collection of images with annotated bounding boxes that indicate head positions. The images are mirrored to increase the sample size. For each image, the considered annotated bounding boxes are used to extract positive training examples. Negatives are sampled randomly across the image as long as they do not overlap too much with annotations (regardless of whether the annotation is considered or ignored).

After collecting the training examples, libSVM [Chang and Lin, 2011] is used to train a support vector machine. The penalty multiplier C was chosen to be 10, but other values such as 1 or 100 did not make a difference other than changing the necessary training time.

After the initial training, the classifier is used to detect heads on the images. Detections that do not match with an annotation are false positives, which are used as additional, strong, negative examples for the classifier training. With an increasing number of those bootstrapping rounds, the classifier converges towards a classifier that is trained on all available samples, but with a more reasonable training time. For SVMs there should be at least two rounds of bootstrapping [Walk et al., 2010]. Because of memory constraints, the number of negative training examples is limited to 30000. If there are more than that, then the examples with the smallest scores are removed in order to retain the strongest negatives.

With only a few hundred positives and tens of thousands negatives, the training dataset is unbalanced. To counteract this unbalance, weights are assigned to the samples, so classification errors of the positives are punished harder than for negatives. The sum of the weights is the same for both classes. The computation of the logistic parameters however does ignore those weights. Because of the much higher number of negatives, the probability of a sample being considered negative is much higher than being considered positive, which is reflected especially in parameter b, which shifts the logistic function towards positive scores. This is irrelevant for the head detection, as probabilistic output is not needed, but might be problematic for the tracking framework. An ad-hoc solution is to set parameter b to zero, so that samples lying on the decision hyperplane (SVM score of zero) are assigned a probability of 0.5.

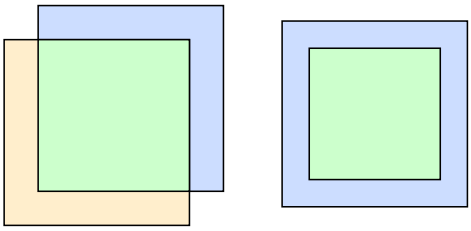


Figure 4.5: Examples of bounding boxes whose overlap ratio is 50 %.

4.5 Experiments

The experiments analyze which feature variations are more suitable for head detection and whether it is possible to reliably detect heads of different orientations using the proposed detector. As opposed to face and pedestrian detection, there is only little literature and no published code on head detection. Therefore, the experiments do not feature a comparison to the state of the art.

4.5.1 Methodology

The detectors are evaluated using a method similar to [Everingham et al., 2010] and [Dollár et al., 2012]. It uses the bounding boxes with their associated scores, which is the result of applying the detector to an image. The detections are greedily matched with the annotated bounding boxes, beginning with the highest scoring detection. Annotations and detections can only be matched once.

The Pascal criterion [Everingham et al., 2010] with an overlap ratio threshold of 50 % is used to determine whether a detection matches an annotation. The overlap ratio is the intersection over union of the predicted and annotated bounding box. See figure 4.5 for examples of bounding boxes that have an overlap ratio of 50 %. If there is more than one matching annotation to a detection, then the one with the higher overlap is used.

Before the matching, the aspect ratio of all annotations is adjusted, as it is important for it to match the aspect ratio of the detections [Dollár et al., 2012]. This is done by increasing either width or height, so the original annotated bounding box is completely and narrowly enclosed by the new boundary.

The evaluation then goes as follows: Detections matched to a considered annotation count towards true positives, detections matched to no annotation count towards false positives, detections matched to ignored annotations are ignored (neither true, nor false positive), unmatched considered annotations are false negatives, and unmatched ignored annotations are ignored (neither true, nor false negative). With this evaluation method, a detector trained and evaluated on the frontal class is not punished for detecting a head that is barely in profile view. In the end, the aim is to detect heads and not just a subset, even if only a subset is used for training or testing.

By varying the threshold of the SVM, a different number of detections is reported for an image. A lower threshold leads to more false positives and less misses, whereas a higher threshold leads to more misses and less false positives. These results are plotted in a detection error tradeoff (DET) curve with false positives per image (FPPI) on the abscissa and miss rate on the ordinate. When a single value representing the performance of a detector is needed, the log-average miss rate (LAMR) is used, as introduced by [Dollár et al., 2012]. It is the average of the miss rates at nine evenly spaced FPPI rates in log space between 10^{-2} and 10^{0}.

To compensate for the rather small dataset, 4-fold cross-validation is used for the evaluation, so each image is used for training as well as testing.

In addition to the detection performance (miss rate at different FPPIs), the efficiency is measured in frames per second (FPS). It is computed by dividing the number of images by the combined detection time. This includes the non-maximum suppression, which has more work to do compared to a real application, because the SVM threshold is set to -1 in order to have many detections and generate a longer DET curve. This does not effect the processing time too much though, as feature computation and convolution are much more expensive. The resulting FPS values are based on the dataset and hence will be different from the final application, but it enables a comparison of the detectors.

4.5.2 Bootstrapping

The training of a frontal head detector is used to determine the training parameters. One important factor is the bootstrapping procedure. For person detection using an SVM, there should be at least two bootstrapping rounds [Walk et al., 2010].

BS #	strategy 1		strategy 2		strategy 3	
	mean	deviation	mean	deviation	mean	deviation
0	26.72 %	1.51 %	26.72 %	1.51 %	26.72 %	1.51 %
1	15.98 %	0.39 %	24.84 %	0.96 %	18.00 %	1.04 %
2	16.48 %	0.84 %	15.54 %	0.29 %	14.04 %	0.26 %
3	15.96 %	0.39 %	14.49 %	0.18 %	13.70 %	0.14 %
4	15.30 %	0.29 %	14.67 %	0.33 %	13.77 %	0.11 %
5	15.29 %	0.67 %	14.21 %	0.31 %	13.77 %	0.16 %

Table 4.1: Mean and standard deviation of the log-average miss rate with a varying number of bootstrapping rounds. *Strategy 1:* Up to 20 hard negatives per image with a score threshold of 0. *Strategy 2:* Up to 20 hard negatives per image with a score threshold of -1. *Strategy 3:* Up to 100 hard negatives per image with a score threshold of -1.

The first strategy starts with 20 random negatives per image in the very first training round and adds up to 20 hard negatives per image in the following bootstrapping rounds. A varying number of those rounds are tested against each other regarding average performance and its deviation over five runs. As can be seen from table 4.1, the average performance increases (log-average miss rate decreases) for strategy one.

Usually, hard negatives are defined as false positives, which in the case of SVMs are negative training samples with a positive score. The decision hyperplane of an SVM however is influenced by training data within the margin, which is defined as the space around the decision boundary with a maximum distance of one. Therefore, including negative training examples with scores above -1 might be advantageous, which is strategy two in table 4.1. The mean and variance of the performance are better in this case. The only interesting observation is the performance with only one bootstrapping round, which does not seem to improve much compared to no bootstrapping at all.

Because there are much more hard negatives now, the maximum number of 20 per image is quite limiting. Therefore, in strategy three, it is increased to 100 per image. This improves the performance even more, as can be seen in the last two columns of table 4.1. The best results are achieved with three rounds of bootstrapping, after that there is not much change.

Thus, for all the following trainings, the procedure goes as follows: Begin with choosing 20 negative examples randomly per image, train the classifier using the negatives and annotated positives, follow that up with three bootstrapping

rounds that add up to 100 hard negatives per image, which are defined as image patches with a score over -1 and an overlap ratio with annotated bounding boxes of at most 0.3. This is an interesting difference to [Walk et al., 2010] and others, that use a threshold of 0 and argue that the amount of strong negatives per image does not have much of an influence.

4.5.3 Window size

An important choice regarding the features is about cell size and number of cells within the detection window. Cell sizes between four [Dollár et al., 2010] and eight pixels [Dalal and Triggs, 2005] are common. For the detection of heads, a square window size is appropriate, as bounding boxes of profile heads are approximately square. Frontal heads are typically a bit more narrow, but at the end of the day the detection of heads in all orientations counts, so an aspect ratio of 1:1 is chosen.

The smallest detectable head size in an image without upscaling is a constraint when choosing cell and window size. It depends on the camera, image resolution, and distance at which humans should be detected. Based on the parameters of the omni-directional image, a size of 40 pixels and above is chosen as a bare minimum of which heads need to be detectable. Therefore, the detection window must not exceed this size, as otherwise the detector could not find heads with a height of 40 pixels (without upscaling). This is also the reason for heads of the dataset smaller than that to be ignored rather than considered. To make the detector performances comparable, they are restricted to find bounding boxes down to this minimum height, but not smaller. Thus, there is no advantage or disadvantage for detectors that are capable of finding smaller heads.

The following combinations of cell and window sizes are tested against each other on the frontal head class:

3x10: Cell size of 3 pixels, window size of 10x10 cells (30x30 pixels).
3x11: Cell size of 3 pixels, window size of 11x11 cells (33x33 pixels).
3x12: Cell size of 3 pixels, window size of 12x12 cells (36x36 pixels).
3x13: Cell size of 3 pixels, window size of 13x13 cells (39x39 pixels).
4x8: Cell size of 4 pixels, window size of 8x8 cells (32x32 pixels).
4x9: Cell size of 4 pixels, window size of 9x9 cells (36x36 pixels).
4x10: Cell size of 4 pixels, window size of 10x10 cells (40x40 pixels).
5x7: Cell size of 5 pixels, window size of 7x7 cells (35x35 pixels).

Size	Height [px]	LAMR	FPS
3x10	30	21 %	29
3x11	33	20 %	20
3x12	36	17 %	9
3x13	39	17 %	9
4x8	32	22 %	42
4x9	36	20 %	30
4x10	40	14 %	21
5x7	35	23 %	48
5x8	40	22 %	34
6x6	36	31 %	63
7x5	35	33 %	71
8x5	40	29 %	63

Table 4.2: Comparison of different window size combinations regarding log-average miss rate (LAMR) and frames per second (FPS) of frontal head detectors.

5x8: Cell size of 5 pixels, window size of 8x8 cells (40x40 pixels).
6x6: Cell size of 6 pixels, window size of 6x6 cells (36x36 pixels).
7x5: Cell size of 7 pixels, window size of 5x5 cells (35x35 pixels).
8x5: Cell size of 8 pixels, window size of 5x5 cells (40x40 pixels).

Table 4.2 shows the log-average miss rate and FPS. The DET curves of four combinations are shown in figure 4.6. A cell size of 4 pixels with a window size of 10x10 cells performs best by some margin.

The results suggest that increasing the window size via either number of cells or cell size also increases the performance. A bigger window captures more details, which might help discriminate heads from everything else, at least up to a certain point. This is not explored any further however because of the constrained window size.

The most time is spent at finding the smallest heads, as the largest image pyramid layers have to be processed. If a detector is restricted to a minimum head size of 40 pixels, but would be able to detect smaller heads, then it does not need to process the largest pyramid layers, but starts at smaller ones. This leads to a faster detection, as can be seen from the table, e.g. by comparing **4x8** with **5x8** or **3x10** with **4x10**.

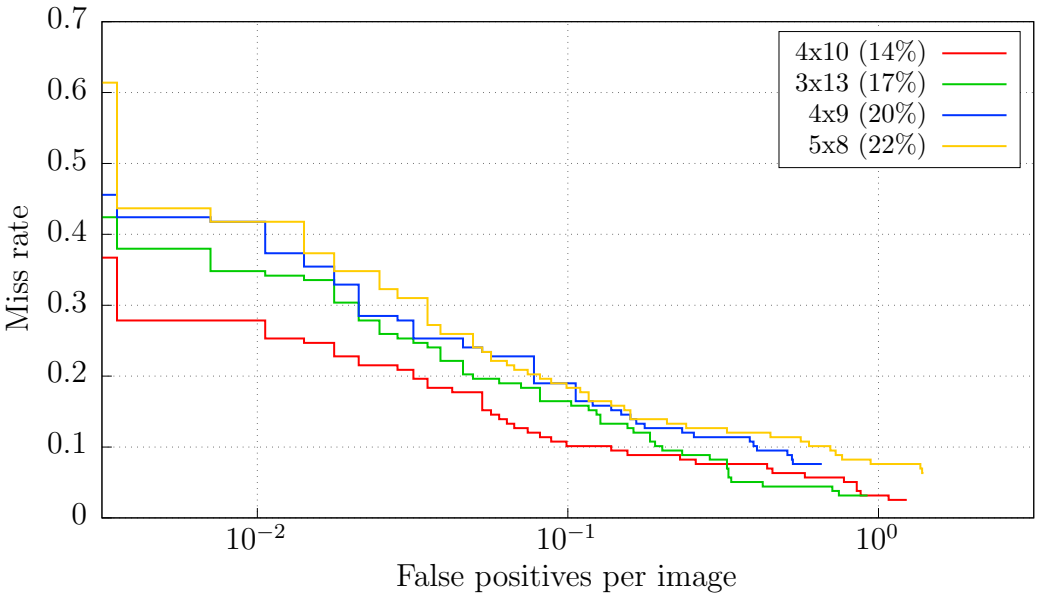


Figure 4.6: DET curves of four window size variations of frontal head detectors. Other sizes performed similarly or worse and were omitted for clarity.

Having a smaller number of cells within the window also improves the efficiency, as there are less computations for each convolution across the feature image. This can be seen by comparing different detectors that all result in windows of the same size, see **4x10**, **5x8**, and **8x5**. All have a height of 40 pixels, but the latter one has the least number of cells and thus is the fastest.

4.5.4 Feature type

After having found a cell size of 4 pixels with a window size of 10 cells to be very promising, different feature parameters are tested to find out whether there is a faster variation that does not perform much worse. The previously chosen features will be called **FHOG9**, as they have nine bins for the unsigned orientation histogram. They are compared against features with less bins and to variations without either of the histogram parts, with only the unsigned one (**UFHOG9**) or the signed one (**SFHOG9**) in addition to the gradient energy features.

It could be assumed that more orientation bins capture more details and thus perform better, but the results in table 4.3 contradict this assumption at least for the chosen feature size. Having only seven (**FHOG7**) or four (**FHOG4**)

Feature	Dim	Size	LAMR	FPS
FHOG2	10	4x10	29 %	44
FHOG3	13	4x10	16 %	38
FHOG4	16	4x10	<u>14</u> %	34
FHOG5	19	4x10	18 %	30
FHOG6	22	4x10	15 %	27
FHOG7	25	4x10	<u>14</u> %	25
FHOG8	28	4x10	17 %	23
FHOG9	31	4x10	<u>14</u> %	21
UFHOG9	13	4x10	28 %	32
SFHOG9	22	4x10	16 %	25

Table 4.3: Comparison of different feature types for frontal head detectors regarding cell descriptor dimensions (Dim), log-average miss rate (LAMR) and frames per second (FPS). The best performances are underlined.

bins performs as well as having nine (**FHOG9**), while the variations in between are slightly worse.

Throwing away the unsigned orientation histogram (**SFHOG9**) has only a small performance impact, while dropping the signed orientation histogram (**UFHOG9**) is much worse. The DET curves in figure 4.7 show more detailed performance comparisons for the six best performers.

The efficiency improvement is easier to guess. The detector has to do one convolution per scaled image for each cell descriptor dimension. Less dimensions thus lead to less convolutions and further to a faster detection. Hence, having less orientation bins improves the detection efficiency. Less bins also speeds up the feature computation a little bit. Throwing away one part of the histogram does not profit from this speed-up and thus is slightly slower than variations with both histogram parts but less orientation bins, even if the cell descriptor dimensions stays the same. This can be seen by comparing **SFHOG9** with **FHOG6** and **UFHOG9** with **FHOG3**.

4.5.5 Head orientation

All of the previous experiments were carried out on the frontal head class, because discriminating them against non-heads is simpler than for the other classes. In the optimal case, all head classes are linearly separable from the

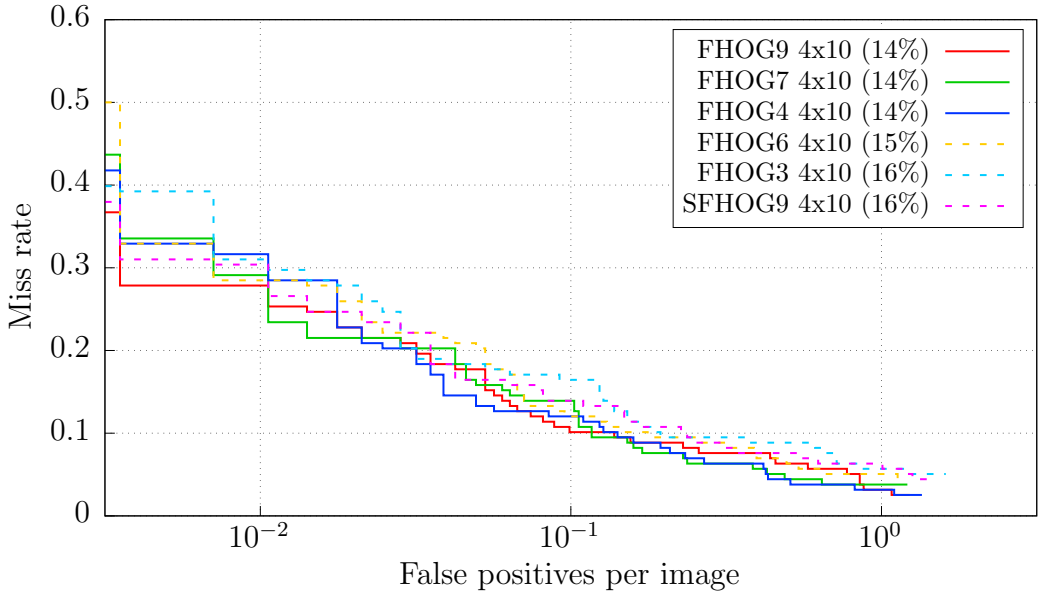


Figure 4.7: DET curves of six feature type variations for frontal head detectors. Other features performed similarly or worse and were omitted for clarity.

background in feature space. Now the interesting question is whether it is possible to detect any kind of head, regardless of pose.

If the harder classes that contain different head poses are not linearly separable from non-heads, they will probably either have less detections or more false positives than detectors trained on a more constrained class. In this case, there is a chance that a detector trained on a subset of heads also performs better on all heads.

The effect of training with different classes using the features **FHOG9 4x10** can be seen in table 4.4. Unsurprisingly, specialists trained on narrow classes perform better on those classes compared to classifiers trained more broadly. This is especially the case for the frontal class, which performs very well for these features.

When tested on all heads, there is not as much difference between the training classes, except for profile. One thing to keep in mind though is the distribution of head classes: almost 40 % of the heads are frontal, while profile and other are around 30 % each. This is an advantage for detectors that were trained more narrowly on the frontal head class but tested on a broader class.

The profile specialist does not excel on its own class as the frontal head detector does. The latter one profits from the symmetry of heads, which

Trained on class	Tested on class			
	Frontal	Profile	Other	All
Frontal	14 %	72 %	79 %	51 %
Profile	72 %	52 %	74 %	66 %
Frontal+Profile	37 %	58 %	72 %	54 %
All	40 %	57 %	65 %	53 %

Table 4.4: Comparison of the log-average miss rate of detectors trained on different classes. Features used are **FHOG9** with a window size of **4x10**. The best performance in each column is underlined.

allows to double the amount of training data by mirroring the examples. The profile head detector on the other hand is trained with heads from both views, the left and the right, although these views differ quite a bit due to not being symmetric.

Figure 4.8 looks closer into the performance of the detectors when tested against all heads. The log-average miss rates are fairly equal for the detectors trained on Frontal, Frontal+Profile (F+P) and All, but the curves have different slopes. The detector trained on the frontal class misses less heads at a small false positive rate, but does not improve as much when increasing the false positive rate. The detectors trained on broader classes improve more and surpass it at around 10 % false positives per image. Other feature variations show a similar behavior regarding the slope, but the curves may be shifted, so the overtaking point is different or the detector trained on all heads performs worse than F+P.

Example images of detections can be seen in figure 4.9. At the default SVM threshold, the detector trained on all heads has less misses and more false positives than a detector trained on frontal heads only. The emphasis on contour leads to wrong detections on round objects with strong edges, as can be seen from the third image in figure 4.9.

The weight vectors of the four SVMs are visualized in figure 4.10 via their unsigned gradient orientation histograms. The frontal SVM reveals a head-like shape and some of the gradients within the head look like eyes, nose, and mouth. This explains why it works well on frontal heads without finding many false positives or heads of other orientations.

In contrast to that, the weight vectors of the other SVMs do not show these characteristics as clearly and look more random. Except for the top part,

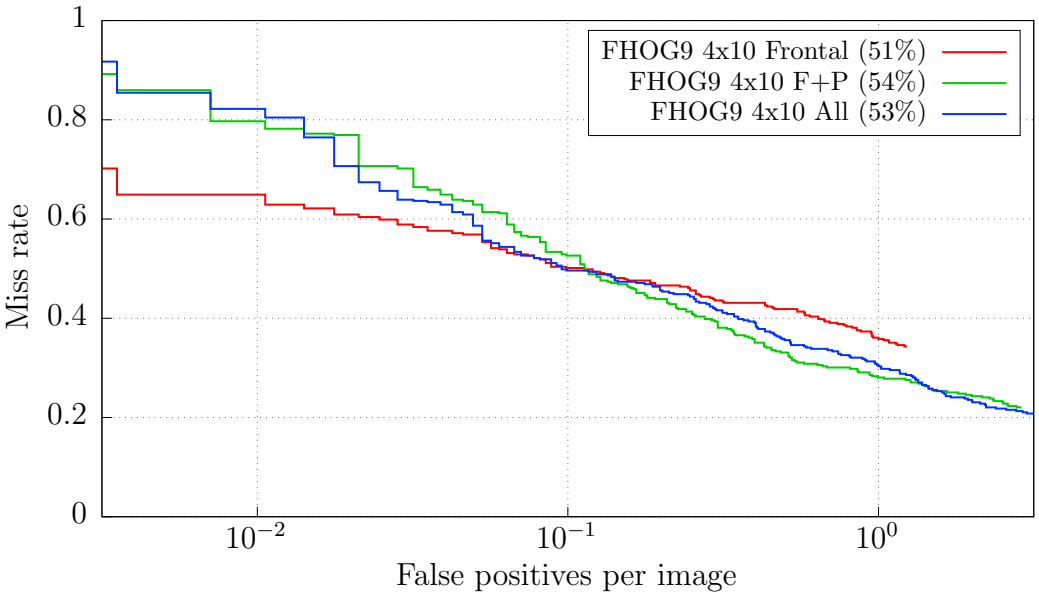
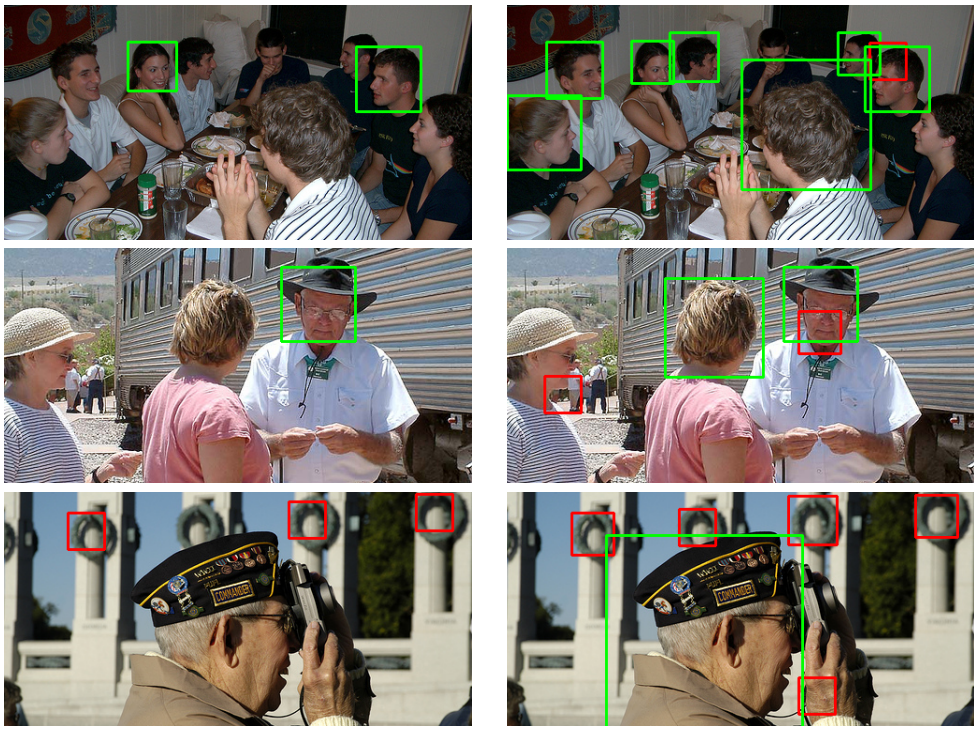
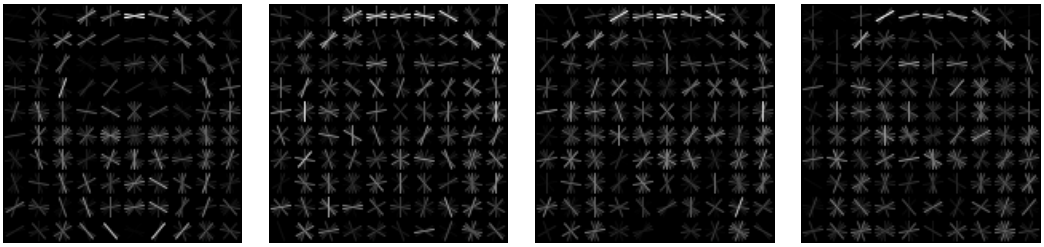


Figure 4.8: DET curves of **FHOG9 4x10** head detectors trained on different classes and tested on all heads.



(a) Detector trained on frontal heads. (b) Detector trained on all heads.

Figure 4.9: Examples of detections with the default SVM threshold of 0.0.



(a) Frontal (b) Profile (c) F+P (d) All

Figure 4.10: Visualization of the unsigned gradient orientation histograms of the positive weights of the SVMs.

Features	Trained on class		
	Frontal	Frontal+Profile	All
FHOG4 4x10	52 %	52 %	65 %
FHOG4 4x9	52 %	60 %	91 %
FHOG4 4x8	52 %	62 %	79 %
FHOG4 5x8	55 %	55 %	75 %
FHOG4 5x7	58 %	85 %	85 %
FHOG9 4x10	51 %	54 %	53 %
FHOG9 4x9	55 %	52 %	52 %
FHOG9 4x8	55 %	57 %	61 %
FHOG9 5x8	56 %	50 %	57 %
FHOG9 5x7	54 %	51 %	59 %

Table 4.5: Comparison of the log-average miss rate of different detectors when tested on all heads.

the head's contour is only barely recognizable compared to the frontal SVM, despite being the most promising feature.

Table 4.5 shows the log-average miss rates of different feature type and training class combinations when tested on all heads. Regardless of the chosen features, the LAMR cannot be improved below 50 %.

4.5.6 Robot camera images

The previously mentioned experiments use images of fairly good quality for training as well as testing. The omni-directional image of the robot on the

(a) Sequence 1: Single person (b) Sequence 2: Three persons

Figure 4.11: Example images from the annotated test sequences.

other hand is blurry and has low resolution. In the final experiment, a small selection of frontal head detectors is trained on all images of the dataset and tested on two image sequences taken with the robot's front-facing camera. The resolution of 752x480 pixels is different from the omni-images, but it shows the same blurriness. It was chosen because it does not have stitching artifacts, which would have made the annotation and evaluation process unnecessarily complicated.

The first sequence (see figure 4.11a) shows a single person in front of a monotonous background. It features different facial expressions, partial occlusions, and several out-of-plane rotations. The second sequence (see figure 4.11b) is more challenging, as three persons occlude each other frequently in front of a more distracting background. It also features lots of out-of-plane rotations. Compared to the training images, the diversity for this small dataset is low, as all persons are fairly young, white, and do not have headdresses. The heads are annotated in each frame by axis-oriented bounding boxes, analogous to the head image dataset used for training. Heads that are more occluded than visible are ignored, meaning they will neither count as hit, nor as miss.

Figures 4.12 and 4.13 show the DET curves of the tested detectors for both sequences. A summary of the values can be seen in table 4.6. The detector with features **FHOG9 4x10** has the best performance on sequence 2 and looks promising overall, but is also quite slow. As constrained computational resources are an issue, a smaller feature vector that leads to a faster detection is beneficial. A good compromise between fast detections and acceptable performance at low false positive rates is **FHOG6 4x8**.

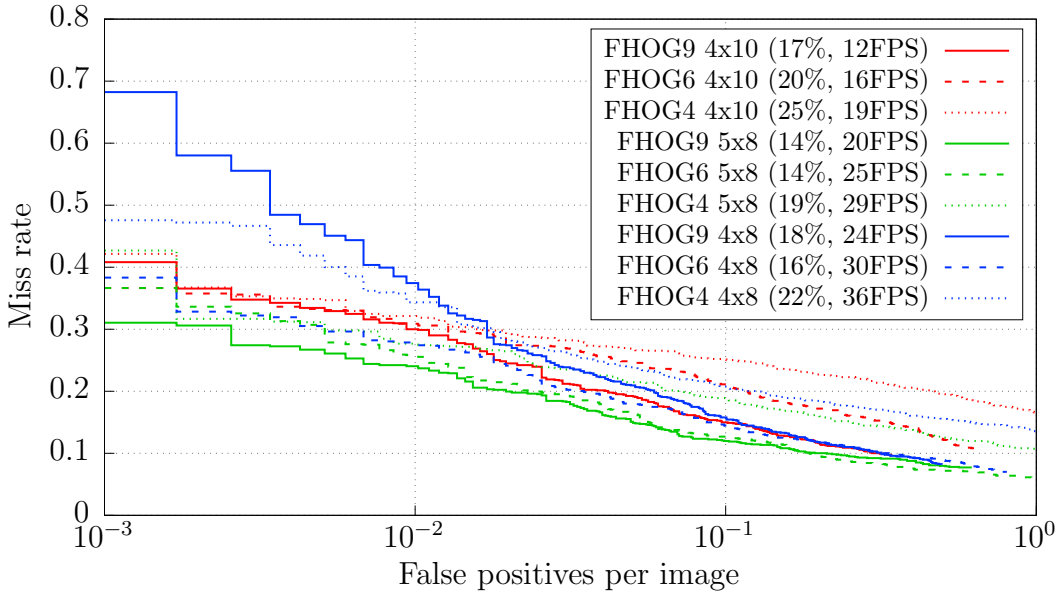


Figure 4.12: DET curves of head detectors tested on sequence 1. Log-average miss rate and frames per second in parentheses.

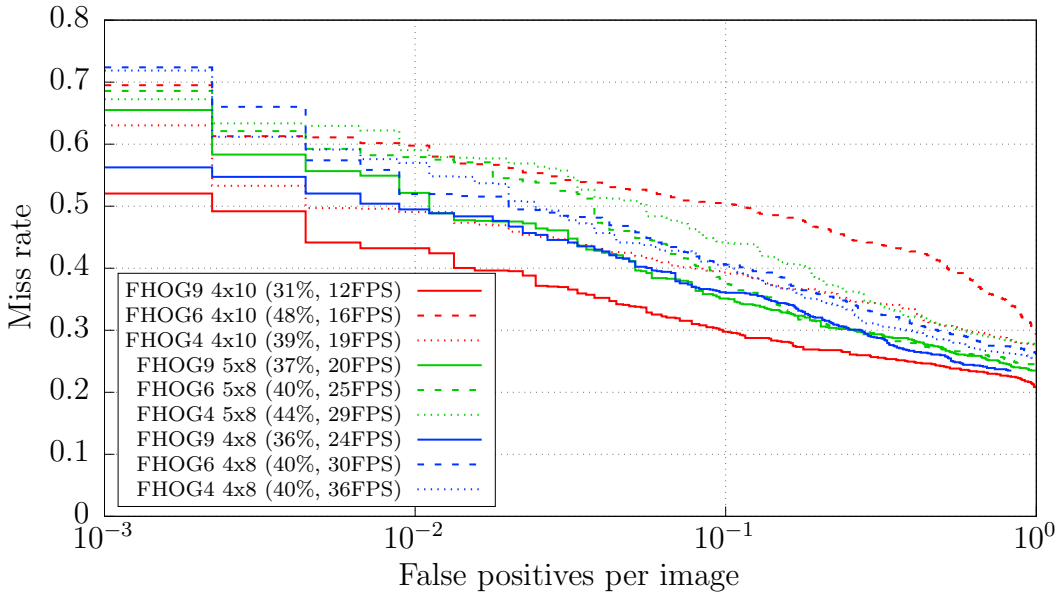


Figure 4.13: DET curves of head detectors tested on sequence 2. Log-average miss rate and frames per second in parentheses.

Features	FPS	MR @ 1% FPPI	
		Sequence 1	Sequence 2
FHOG9 4x10	12	30 %	43 %
FHOG6 4x10	16	31 %	60 %
FHOG4 4x10	19	32 %	49 %
FHOG9 5x8	20	24 %	52 %
FHOG6 5x8	25	26 %	60 %
FHOG4 5x8	29	28 %	59 %
FHOG9 4x8	24	37 %	49 %
FHOG6 4x8	30	28 %	52 %
FHOG4 4x8	36	34 %	57 %

Table 4.6: Comparison of detectors on sequence 1 and 2 regarding frames per second and miss rate at 1 % false positives per image.

4.6 Conclusion

With the chosen features (FHOG) and classifier (linear SVM), it is possible to build detectors[1] that find heads with a log-average miss rate of around 50 %. They mostly differ in what kinds of heads they can detect reliably. If trained on only upright frontal heads, then these can be found pretty well, but others are missed, while a detector trained on a broader class of heads is worse on the frontal ones, but finds more heads of other orientations. Overall, the chosen approach of detecting heads regardless of pose is fast, but leaves room for improvement regarding miss and false positive rate.

A solution is to train specialists and combine their scores, but this would increase the time needed for detection. Having two detectors for profile heads, one for left and one for right, would probably also help. Hence, for a reliable pose invariant head detector, a more sophisticated classifier (e.g. kernel SVM) or a combination of specialized classifiers is necessary. This increases the computational costs, though.

Neither of those approaches is sufficient, as the head detector must be reliable and fast. Of the three requirements, pose invariance is the least important. Choosing the detector that is specialized on the frontal class is beneficial for the application scenario, because persons looking at the robot are more likely to interact with it and thus are more important to detect than others.

[1] The source code of detector and trainer, few trained SVMs, and head annotations that were used for training can be found at https://github.com/ex-ratt/AdapTrack.

A low false positive rate is desired to prevent the robot from looking at persons that do not exist. The detector misses much less heads at higher false positive rates, which enables the frontal head detector to even find some heads of other orientations. A tracker might be able to exploit this fact and combine the best of both worlds, leading to an approach that is fast, reliable, and pose-invariant once initialized on a head. It could be further enhanced by an adaptive short-term tracker, which helps to bridge the gap between successful detections.

Chapter 5

Adaptive Tracking

The previous chapter showed the creation of a head detector that finds heads quite well as long as the face is visible, but shows worse performance from other viewpoints. A tracker that merely connects detections across frames to determine trajectories might thus lose heads that look away from the robot. This chapter is about adaptive model-free short-term tracking, which could bridge the gap between successful detections.

A tracker that is model-free has not been trained off-line with examples of the targets to track. The only supervised training examples are given in the initial frame of an image sequence, e.g. by a bounding box around the target. Short-term tracking refers to the fact that there is no re-detection capability once the target is lost. Adaptive trackers update the target's model after the first frame to adapt to changing appearances and conditions.

In general, the tracker builds an initial model of the target in the first frame and localizes it in subsequent frames. Without adapting the model to new appearances, the target might get lost, but each adaptation may introduce errors, which leads to drift. This is known as the template update problem [Matthews et al., 2004]. The main question of adaptive tracking is thus how and when to update the model to minimize template drift while still maintaining adaptivity.

Besides having little drift, the tracker must support scale changes and full out-of-plane rotations, and it must be very efficient to support multi-target tracking without slowing the system down by much.

5.1 State of the art

The three main components of visual adaptive tracking are target modeling, localization, and updating. Modeling is about finding a representation of the target appearance that helps localizing it in subsequent frames. Localization is about determining the new target position in the next frame, which must be accurate enough to ensure a correct model update. Updating is concerned with incorporating the new target appearance into the model, so tracking can continue even when encountering changing conditions and appearances, e.g. caused by different poses or illumination. See [Salti et al., 2012, Wu et al., 2013, Smeulders et al., 2014] for surveys on visual adaptive tracking.

5.1.1 Model

There are three types of appearance models: generative, discriminative, and hybrids, which combine the former two.

Generative models only rely on the target appearance and its variations. They do not care about the background and other objects, which makes them potentially simpler and more efficient than other models. Examples are templates consisting of pixel values [Matthews et al., 2004], mixtures of simple templates [Kwon and Lee, 2010], histograms [Liu and Sun, 2009], or subspaces based on eigenvectors [Ross et al., 2008]. Instead of having one holistic model for the whole target, it can also be broken down into parts. This increases the robustness against partial occlusions and pose changes [Adam et al., 2006].

Discriminative models describe a decision boundary in feature space that distinguishes the appearance of the target from that of the background and other distracting objects. This leads to a better tracking performance when background or other objects have a similar appearance to the target. Often, binary classifiers and learning algorithms like support vector machines [Tang et al., 2007, Supančič III and Ramanan, 2013], random forests [Santner et al., 2010], or boosting [Grabner et al., 2008, Klein and Cremers, 2011] are involved. ALIEN [Pernici and Del Bimbo, 2013] uses two nearest neighbor classifiers, one for the tracked object containing feature points with their descriptors, and one for the context and background containing descriptors of points that do not belong to the tracking target. The approaches of [Li et al., 2008] and [Stalder et al., 2009] use three discriminative classifiers. One of them

is trained off-line and serves as detector and prior to prevent drift, which makes those approaches not model-free. The other two classifiers are trained on-line. One of those is highly adaptive, while the other is updated more conservatively.

A similar combination of multiple models with different life-spans is PROST [Santner et al., 2010], where a simple template (generative) is not adapted anymore after the very first frame, optical flow (generative) is used as a highly adaptive tracker, and a random forest classifier (discriminative) is in between. This is an example of a hybrid model that combines the generative and discriminative approaches. Optical flow is used in several trackers as a highly adaptive component, which is also able to detect tracking failures and occlusions, thereby preventing erroneous updates of the discriminative classifier [Kalal et al., 2012, Hua et al., 2014]. Other approaches use and update the generative and discriminative model equally to create a more robust combined model [Yu et al., 2008, Zhong et al., 2012].

Besides the model, another important choice is which features to extract. Raw intensity values are susceptible to illumination changes, while normalized intensity histograms are not. Extracting features that are invariant to some condition or pose changes makes tracking simpler, as less adaptivity and updates are necessary to sufficiently describe the target appearance.

Features that describe texture or are based on raw intensity or color values capture the surface of an object, as do feature point based models. In the case of heads, a tracker would be more likely to stick to the face than to follow the head, which leads to a tracking failure or drift in case of severe out-of-plane rotations [Kalal et al., 2012]. Features that describe edges and gradients on the other hand are able to capture the contour of objects, which is beneficial when tracking heads under pose changes.

The appearance models of all of the discussed trackers are two-dimensional, which comes from the fact that the image is a two-dimensional projection of the scene. Usually, the target to be tracked is three-dimensional, though. A 3D-model, e.g. based on feature points [Lebeda et al., 2014] or planar surfaces [Xiang et al., 2014], does not need to be invariant against any rotation as long as the object's orientation is estimated in addition to its position. This makes tracking more computationally expensive, though, and becomes impractical when tracking multiple objects.

5.1.2 Localization

Visual tracking usually involves finding the location that responds the most to the appearance model. Sliding-window techniques scan across the whole image [Tang et al., 2007, Supančič III and Ramanan, 2013] or a local region around the previous target position [Grabner et al., 2008, Babenko et al., 2011] to find the maximum response. Searching across the whole image helps when there is severe camera motion and enables re-detecting the target after an occlusion or tracking failure, while having a smaller area to search results in less computational demands.

Some approaches do not consider scale changes, assuming the target to have a fixed size [Hare et al., 2011]. A sparse template of the target appearance can lead to a smooth response map, allowing an efficient logarithmic search to find the position that best matches the template [Kolarow et al., 2012]. A part-based model allows for individual responses of each part, which in combination with a least-median-squares estimator increases the robustness against partial occlusions [Adam et al., 2006, He et al., 2013].

The new target location can simply be given by the response map's maximum [Bolme et al., 2010, Hare et al., 2011, Henriques et al., 2015]. Particle filtering approaches [Liu and Sun, 2009, Klein and Cremers, 2011] can work with multiple modes in the response map and usually include the target dynamics when determining the new position.

Optical-flow based methods use the motion estimates of many tracked points to determine the target location [Vojíř and Matas, 2014]. Some approaches combine an optical-flow motion estimate with the position determined by a detector. PROST [Santner et al., 2010] uses a rule-based combination, where one estimation can overrule the other if certain conditions are met, while TLD [Kalal et al., 2012] lets a conservative classifier decide which of the estimates to trust. [Poschmann et al., 2014] use optical flow to guide the samples of a particle filter, while an additional sliding-window detector can reset the particle filter if a highly confident detection does not sufficiently overlap with the filter's estimate.

Another distinction between different trackers is whether they assume the target to be visible in every frame. If they do, the best matching target position is reported for each frame, regardless of its quality or certainty [Adam et al., 2006, Henriques et al., 2015]. Highly adaptive short-term trackers, like optical-flow based methods [Vojíř and Matas, 2014], usually

operate under this assumption, as they do not maintain an appearance model for more than one frame.

This visibility assumption is beneficial for abrupt appearance changes, as occlusions are not considered to be the cause and tracking continues, but is certainly disadvantageous in case of full occlusions, as the tracker will start to follow some other object. Disappearance and occlusion can be detected by e.g. a low classifier score [Tang et al., 2007, Klein and Cremers, 2011], a low peak-to-sidelobe ratio in case of correlation trackers [Bolme et al., 2010], a high number of failing optical flow trackers [Kalal et al., 2012], or a high error when matching templates [Kolarow et al., 2012].

5.1.3 Update

Updating the model of an adaptive tracker can be seen as a semi-supervised learning problem, where few initial training examples are labeled in the first frame, and many additional unlabeled training examples are available from the following frames. The task then is to train a classifier with labeled as well as unlabeled examples.

In self-learning, a classifier predicts the labels of the new training data for itself, which leads to mistakes reinforcing themselves. A more robust approach is co-training [Blum and Mitchell, 1998], which uses two sufficient and independent views, each with its own classifier, that predict the labels for each other. This method was used successfully in adaptive tracking [Tang et al., 2007, Yu et al., 2008]. Other examples of trackers with semi-supervised training paradigms are based on multiple instance learning [Babenko et al., 2011, Leistner et al., 2010] or semi-boost [Grabner et al., 2008, Liu and Sun, 2009]. STRUCK [Hare et al., 2011] avoids the intermediate binary labeling and classification step entirely by directly learning the displacement using a structured output SVM, but can only estimate translations and not scale changes.

Semi-supervised learning algorithms make assumptions about the training data, e.g. examples that are close to each other are likely to share the same label. Similarly, two key assumptions in tracking help to generate labels for the new training examples: The target moves along a trajectory (temporal structure) and is unique within the frame (spatial structure) [Kalal et al., 2012]. If the target moves along a trajectory instead of randomly jumping around, then a tracker is able to follow it from one frame to another and give a positive label to the newly found position. If the target is unique,

then everything else besides the found position can be regarded as having a negative label. Many trackers use these assumptions to sample new training data. Negatives might be generated in a deterministic manner around the target position [Liu and Sun, 2009], or by selecting peaks of the response map other than the target, e.g. from a sliding-window detector [Tang et al., 2007] or random sampling [Klein et al., 2010].

This procedure of generating new training data has similarities to self-learning, as the tracker trains itself, and small errors add up. An inaccurate localization leads to a suboptimal positive training example, which in turn will have a negative influence on the next localization, and so on. This is especially true for highly adaptive trackers whose model is rebuild with just the appearance information of the current frame, so their localization has to be very precise to reduce drift [Kolarow et al., 2012, Vojíř and Matas, 2014]. Not adapting at all, on the other hand, cannot drift, but might lose the target in case of new appearances, or has to rely on the invariance of the extracted features [Adam et al., 2006].

These examples are the two extremes of the stability-plasticity dilemma [Grossberg, 1987]. Applied to tracking, it states that the model should adapt to significant new target appearances, but at the same time should not forget past appearances and remain unchanged in case of irrelevant appearance changes, e.g. caused by occlusion or inaccurate localization.

Because of processing time requirements, the model cannot be arbitrarily complex, so usually there is a compromise between keeping old and adding new information. [Matthews et al., 2004] keep the first and last template, where the initial one is used for the final localization to prevent drift. A similar idea is to never remove the initial positive sample and only replace the positives that the classifier is most confident about [Klein et al., 2010]. Another replacement strategy is to simply discard the oldest examples [Tang et al., 2007]. Removing the support vector which results in the smallest change of the SVM weight [Hare et al., 2011] and merging sub-spaces [Yu et al., 2008] are other strategies to restrict model complexity. By linearly interpolating between the old and new model of the current frame, correlation trackers have some memory, where the appearance of the previous frames decays exponentially over time [Bolme et al., 2010, Henriques et al., 2015].

Trackers that assume the target to be visible in every frame typically also update the appearance model in every frame. Not all of these updates are desired though, as occlusions and tracking failures may happen. These are detected just like the visibility of the target, see sec. 5.1.2, possibly with

different thresholds. The addition of such learning conditions can improve the tracking performance [Klein and Cremers, 2011, Poschmann et al., 2014].

All of the described adaptation mechanisms update the model immediately after determining the target position and cannot correct mistakes other than by forgetting. Maintaining multiple models in parallel, each generated from a different potential target trajectory, allows to correct mistakes by choosing a model and trajectory that best explain the new observation. This approach is explored by [Khan et al., 2004] using a Rao-Blackwellized particle filter, where each particle builds its own generative model. This increases the computational complexity however, especially for discriminative models whose update is usually not as fast.

5.2 Approach

The adaptive short-term single-target tracker is designed such that its model and update strategy can easily be integrated with a head detector and tracker. The localization on the other hand is only necessary to provide a stand-alone application for the experiments.

5.2.1 Model

The target appearance model is based on a linear support vector machine (SVM) and FHOG features. When combined with the head detector, the tracker could easily reuse its features. Benefits of using a linear SVM are the quick classification of feature vectors and good discrimination between tracked target and everything else in its surroundings.

To keep tracker and detector similar, the feature type is with **FHOG6 4x8** largely identical. The only difference occurs in case of non-square aspect ratios at initialization, as either the width or height may be less than eight cells. After (re-)initialization, the aspect ratio remains fixed and is not adapted.

5.2.2 Localization

The new target position is determined by searching the local neighborhood around the previous location for the highest scoring position candidate. To

achieve sub-cell accuracy, the position is further refined by considering the scores of surrounding candidates. Scale changes are supported by two additional search windows, where one is at a larger scale and the other at a smaller one.

5.2.3 Update

The appearance model is updated after estimating the target position. This assumes that the change in appearance between two frames is small enough, so that the previous model can be used to localize the target in the current frame.

Initially, the feature vector of the given bounding box is used as the only positive example, while a low number of negatives is randomly sampled from the surroundings. An initial SVM is trained with this data. Its sole purpose is to collect hard negatives, which are image patches with a high SVM score. These new negative examples are used with the positive example to train a second SVM, which is used for target localization and subsequent updates.

In the following frames, the estimated target position is used to extract a single positive example, and hard negatives are sampled from the surroundings. A new SVM is trained with data from the current frame only, which ensures a fast training. To still provide the model with a memory, the parameters of the new and old SVM are linearly interpolated according to a learn rate λ:

$$\begin{aligned} \mathbf{w}_k &= \lambda \mathbf{w} + (1 - \lambda)\mathbf{w}_{k-1} \\ \rho_k &= \lambda \rho + (1 - \lambda)\rho_{k-1}, \end{aligned} \tag{5.1}$$

where k is the index of the frame, $\mathbf{w}_{k-1}$ and ρ_{k-1} are the parameters of the current SVM, $\mathbf{w}$ and ρ are the parameters of the newly trained SVM, and $\mathbf{w}_k$ and ρ_k are the interpolations which are used to locate the target in the next frame.

5.3 Experiments

The experiments show how the learn rate parameter influences the tracking performance and how the proposed tracker compares to the state of the art. Dataset and benchmark are provided by the toolkit of the annual visual object tracking (VOT) challenge [Kristan et al., 2016a].

5.3.1 Methodology

A tracker is initialized on the first frame of an image sequence and predicts the target position in the following frames. The bounding box overlap ratio is determined in each frame and counts towards the accuracy of the tracker. A higher average overlap ratio means that the tracker followed the target more closely and is thus more accurate.

An overlap ratio of zero indicates a tracking failure. In such a case, the tracker is re-initialized after skipping five frames. The number of failures counts towards the robustness of the tracker. A lower number of failures means that the tracker is more robust. The accuracy measurement only starts ten frames after an initialization to reduce the influence of re-initialization on the accuracy.

The performance of competing trackers is represented by a point on an accuracy-robustness plot (AR plot). The robustness $R_S = e^{-SM}$ is the probability of successfully tracking a target for at least S frames after the last failure. It is computed from the mean-time-between-failures $M = \frac{F}{N}$, where F is the number of failures and N is the length of the sequence.

The expected average overlap is a third measure, which estimates the average overlap of a tracker on sequences with similar properties to the dataset. The efficiency of the trackers is measured in effective filter operations, which results from dividing the measured tracking time by the time required for a pre-defined filtering operation, thus reducing the influence of hardware.

Stochastic trackers are tested 15 times on each sequence and the overlap ratios and failure rates are averaged. Details about the evaluation methodology and measures can be found in [Čehovin et al., 2016, Kristan et al., 2016b, Kristan et al., 2015].

5.3.2 Trackers

The visual adaptive short-term tracker proposed in this chapter is named HOG+SVM tracker (HST) in the following experiments. Different learn rates λ are tested against each other. They range from preventing updates after the initial frame ($\lambda = 0.0$) to disregarding any memory of past frames ($\lambda = 1.0$). The HST variations are compared to a subset of trackers from the VOT challenge 2016 [Kristan et al., 2016a] that are similar in speed.

Flock of Trackers (FoT) [Vojíř and Matas, 2014] robustly combines the transformation estimate of a number of local optical flow trackers. A similar method is Best Displacement Flow (BDF) [Maresca and Petrosino, 2014], which uses clustering to identify the best displacement vector.

The Kernelized Correlation Filter (KCF) [Henriques et al., 2015] is the base for two trackers. The Scalable Kernel Correlation Filter with Sparse Feature Integration (sKCF) [Solís Montero et al., 2015] extends KCF by estimating scale changes using optical flow of relevant keypoints. Additionally, the model is only updated for similar KCF responses, which reduces template drift. Besides scale estimation, the Scale and Motion Adaptive Correlation Filter Tracker (SMACF) also adds colorname features and a first order motion model to KCF.

Scale Adaptive Mean Shift (ASMS) [Vojíř et al., 2014] is a tracker that uses the mean-shift procedure for the Hellinger distance to estimate translation as well as scale change. Additional extensions make it more robust against background clutter and self-similar objects. The last tracker is based on Normalized Cross-Correlation (NCC). It searches for the best match of a grayscale template, which is created on initialization and not updated afterwards.

5.3.3 Results

Performance and efficiency values can be found in table 5.1. Figure 5.1 shows an accuracy-robustness plot of the evaluated trackers.

The learn rate defines how quickly HST adapts. With a value of 0.0, there is no update after the initialization, which prevents any template drift. Therefore, this is the most accurate variation of the tracker. However, the missing adaptation leads to failures when target appearance or background changes. Increasing the learn rate reduces the accuracy, as it introduces some drift, but the robustness is increased up to a learn rate of about 0.1. Higher learn rates further increase drift, which eventually decreases the robustness again.

HST cannot compete with state-of-the-art short-term trackers. They are much more tailored towards model-free tracking, whereas HST is ought to be merely one component of head tracking. It spends most of its time (around 70 to 90 %) on computing the features, which is no longer required when paired with the detector, as the tracker then does not have to compute the features again. This makes it much more efficient than the other trackers when integrated into an application that tracks multiple heads.

Tracker	EAO	Accuracy	Failures	EFO
SMACF	<u>22.6</u> %	<u>50.3</u> %	<u>0.44</u> %	91.5
ASMS	21.2 %	<u>50.3</u> %	0.52 %	82.6
sKCF	15.3 %	48.5 %	0.82 %	91.1
FoT	14.2 %	37.7 %	0.82 %	105.7
BDF	13.6 %	37.5 %	0.79 %	138.1
HST $\lambda = 0.1$	10.5 %	38.3 %	1.08 %	104.5
HST $\lambda = 0.02$	9.8 %	43.4 %	1.31 %	105.7
HST $\lambda = 0.5$	8.7 %	34.4 %	1.30 %	111.5
HST $\lambda = 0.0$	8.6 %	49.6 %	2.02 %	109.9
NCC	8.0 %	49.0 %	2.10 %	<u>226.9</u>
HST $\lambda = 1.0$	6.9 %	31.2 %	1.60 %	113.0

Table 5.1: Expected average overlap (EAO), average overlap (accuracy), failures per frame, and speed (in EFO units) of the evaluated trackers. The best value in each column is underlined.

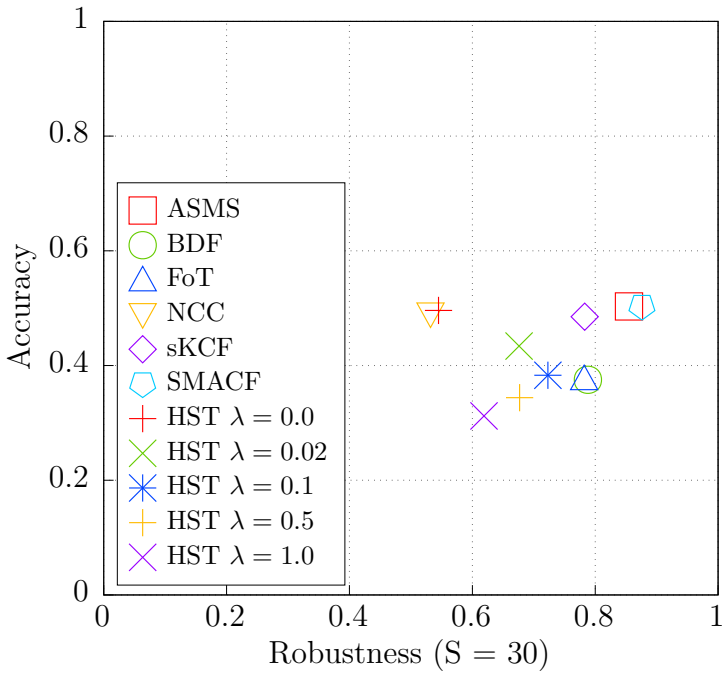


Figure 5.1: Accuracy-robustness plot of the evaluated trackers.

5.4 Conclusion

This chapter introduced a single-target short-term tracker[1] and compared it to the state of the art. It is able to track arbitrary objects, but will generally fail in case of substantial appearance changes. The FHOG features make it well suited for targets with a strong contour, whereas articulated objects and changing aspect ratios often lead to failures. It is not quite on the same level as trackers that are specifically designed for model-free tracking. However, this is not the intention, as it will be used within a head tracking application.

The short-term tracker is very efficient when combined with the head detector, as the features do not need to be computed again. Low accuracy resulting from template drift will be less of a problem, as the head model of the detector is available for tracking. But for an optimal combination, the continuous head classifier confidence must be utilized instead of only final detections.

[1] The source code can be found at https://github.com/ex-ratt/AdapTrack.

Chapter 6

Head Tracking

The previous two chapters were about head detection and adaptive single-target short-term tracking. In this chapter, both are combined to yield a robust head tracker that determines the positions of multiple heads. The detector initializes new tracks. Its continuous confidence density guides the tracker and reduces the drifting problem. The adaptive model specializes on a specific head and background appearance, enabling to track the head even in non-frontal views. The resulting multi-head tracker must not drift by much, cope with imperfect bounding boxes at initialization, and has to detect the disappearance of tracked heads.

6.1 Related work

Similar to head detection, there is not as much literature on tracking heads under full out-of-plane rotations as there is on face or person tracking. Most head trackers model the contour, either using splines [Isard and Blake, 1998, Bouaynaya and Schonfeld, 2005], ellipses [Birchfield, 1998], or HOG features [Benfold and Reid, 2011]. Sometimes this is accompanied by a color histogram [Birchfield, 1998, Bouaynaya and Schonfeld, 2005].

A different approach is to model the head as a texture-mapped cylinder [La Cascia et al., 2000]. The person has to face the camera before tracking can start, because a 2D face detector is used for initialization. The face texture is mapped onto one half of a cylinder. The tracker estimates the 3D position and orientation of the head and was tested with out-of-plane

rotations of up to 60 °. However, there is no mentioning of full out-of-plane rotations or updates of the cylinder's texture after the first frame.

The head tracker of [Birchfield, 1998] can handle significant out-of-plane rotations. It models the head as an ellipse, using the gradient at the border and color histogram within, which includes skin as well as hair color. However, the histogram must be computed off-line using an image with a three-quarters view of the head that is to be tracked.

[Isard and Blake, 1998] use condensation, a particle filter variation, to estimate the contour of the tracked head. This is extended by [Bouaynaya and Schonfeld, 2005] with a color module and post-tracking refinement using an active contour model. Both approaches, as well as the two mentioned before, are single-target trackers.

A closely related work is the multi-head tracker by [Benfold and Reid, 2011], where a head detector based on HOG features and a support vector machine is combined with an optical flow tracker. To achieve reliable results, the data-association procedure needs information from future frames, thus introducing a latency of few seconds between receiving a frame and generating its output. This is not crucial for a surveillance scenario, but in a robotic application, the latency should be as low as possible to allow the robot to react quickly.

6.2 Approach

The proposed visual tracker estimates the state $\mathbf{x} = (x, y, s)^T$ of a head, which describes a square bounding box in image space. The position (x, y) and size s is given in pixels. If the aim was to create a purely visual tracker, then it would make sense to add velocities and an occlusion label to the state. As this chapter focuses on the measurement model and adaptive component, the tracker itself is kept rather simple.

To cope with the potentially multi-modal non-Gaussian measurement likelihood and make use of the detector confidence instead of only final detections [Breitenstein et al., 2011], a particle filter is used to estimate the state probability distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ [Doucet and Johansen, 2009]. Particle filters approximate the distribution using a set of n weighted samples that each

consist of a state instantiation $\mathbf{x}_k^{[i]}$ and an importance factor (aka weight) $\pi_k^{[i]}$ with $i \in \{1, \ldots, n\}$:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{n} \pi_k^{[i]} \delta(\mathbf{x}_k - \mathbf{x}_k^{[i]}), \tag{6.1}$$

where $\delta(.)$ is the Dirac delta function and $\sum_{i=1}^{n} \pi_k^{[i]} = 1$. Using the state transition density as proposal distribution yields a simple variation called bootstrap filter [Gordon et al., 1993]. Given the previous particle set $\{\mathbf{x}_{k-1}^{[i]}, \pi_{k-1}^{[i]}\}$, the particles are updated as follows:

1. Draw n equally weighted particles $\mathbf{x}_{k-1}^{[i]}$ with replacement from the particle set according to probabilities $\pi_{k-1}^{[i]}$
2. Sample updated particles from the proposal distribution

$$\mathbf{x}_k^{[i]} \sim q(\mathbf{x}_k^{[i]}|\mathbf{x}_{0:k-1}^{[i]}, \mathbf{z}_{1:k}) = p(\mathbf{x}_k^{[i]}|\mathbf{x}_{k-1}^{[i]}) \tag{6.2}$$

3. Compute weights $\pi_k^{[i]}$ such that $\sum_{i=1}^{n} \pi_k^{[i]} = 1$:

$$\pi_k^{[i]} \propto \frac{p(\mathbf{z}_k|\mathbf{x}_k^{[i]}) p(\mathbf{x}_k^{[i]}|\mathbf{x}_{k-1}^{[i]})}{q(\mathbf{x}_k^{[i]}|\mathbf{x}_{0:k-1}^{[i]}, \mathbf{z}_{1:k})} = p(\mathbf{z}_k|\mathbf{x}_k^{[i]}) \tag{6.3}$$

The state is estimated to be the weighted mean of the particles $\hat{E}[\mathbf{x}_k] = \sum_{i=1}^{n} \pi_k^{[i]} \mathbf{x}_k^{[i]}$.

6.2.1 Motion model

The target motion is described by a random walk model, where each particle is propagated by adding zero-mean Gaussian noise to the state (particle index is dropped for clarity):

$$\begin{aligned} x_k &= x_{k-1} + s_{k-1} v_{x,k}, & v_{x,k} &\sim \mathcal{N}(0, \sigma_x^2) \\ y_k &= y_{k-1} + s_{k-1} v_{y,k}, & v_{y,k} &\sim \mathcal{N}(0, \sigma_y^2) \\ s_k &= s_{k-1} + s_{k-1} v_{s,k}, & v_{s,k} &\sim \mathcal{N}(0, \sigma_s^2) \end{aligned} \tag{6.4}$$

The noise is scaled by the bounding box size, so the variances σ_x^2, σ_y^2, and σ_s^2 can be chosen independently from the image resolution. Heads closer to the camera appear larger and their movements lead to bigger changes of the bounding box compared to heads further away. This also is reflected by the noise being relative to the bounding box size, but assumes a fixed camera pose, as camera movement leads to position changes independent from the bounding box size or distance to the head.

6.2.2 Measurement model

The particles are guided by the combination of two measurement models, one for the class and another for the specific instance. The class-specific measurement model $p_c(\mathbf{z}_k|\mathbf{x}_k)$ is the same for each particle filter and guides the particles towards head-like regions, thereby reducing the risk of template drift, which increases the accuracy. It is based on the head detector's SVM.

The instance-specific measurement model $p_i(\mathbf{z}_k|\mathbf{x}_k)$ is tied to one filter and represents the actual tracked head and its current appearance. It guides the particles to image regions that look similar to the head, thereby reducing tracking failures caused by unknown appearances, which increases the robustness. It is based on the adaptive SVM of the short-term tracker.

Both measurement models re-use the features of the detector to keep the computational demand of each tracked head low. They are based on the probabilistic support vector machine (SVM) (see eq. 4.2), where the input is the feature vector extracted from the image $\mathbf{z}_k$ and bounding box specified by the state $\mathbf{x}_k^{[i]}$.

The class-specific SVM is trained off-line and remains unchanged, while the instance-specific SVM is trained on-line and adapts to the current target appearance. The logistic function parameters are kept fixed however, as the sample size is too small for cross-validation, which would be necessary to avoid overfitting. Not having to compute these parameters also saves some time on the update.

The two measurement models use the same features and classifier and only differ by the training examples. Hence, they are probably correlated, so their probabilities cannot simply be combined by multiplication as if they were independent. Instead, the models are weighted by exponents that sum to one [Brasnett et al., 2007], which increases the uncertainties of the models.

This is similar to covariance intersection [Julier and Uhlmann, 1997b], where the uncertainty of Gaussian estimates is increased to cope with potentially correlated noise.

There is no clear preference for any of the models, so they are weighted equally (particle index is dropped for clarity):

$$p(\mathbf{z}_k|\mathbf{x}_k) = p_c(\mathbf{z}_k|\mathbf{x}_k)^{0.5} p_i(\mathbf{z}_k|\mathbf{x}_k)^{0.5} \tag{6.5}$$

In case the tracker is used without its adaptive component, the measurement model reduces to (again, the particle index is dropped):

$$p(\mathbf{z}_k|\mathbf{x}_k) = p_c(\mathbf{z}_k|\mathbf{x}_k) \tag{6.6}$$

6.2.3 Initiation and termination

Heads are found in each frame by applying a detector. These detected heads are associated with heads that are already tracked using a greedy nearest-neighbor algorithm based on their overlap ratio. Each detected head that is not assigned will initiate a new particle filter. Its particles are sampled from a Gaussian distribution centered at the detection, where the standard deviations are chosen proportionally to the bounding box size. This newly initialized target will not be reported yet, because it may be a false positive detection. Tracked heads with an associated detection are confirmed and only after this confirmation are reported.

Tracked heads that are yet unconfirmed and do not have an associated detection get terminated. Hence, for heads to be tracked, they have to be detected in two subsequent frames, which removes many false positives.

Persons may disappear, either by leaving the field of view or by being occluded. If a track does not have an associated detection and the SVM score for the estimated bounding box falls below a threshold, then the head is considered to not be visible anymore and the track is terminated.

In some rare cases, the occlusion of one person by another might not be detected and the track might drift to the occluder, which then would be tracked twice. Therefore, when the bounding boxes of two tracks overlap

sufficiently, the one with a lower SVM score is terminated, as this one probably represents the occluded head.

In both cases, the visibility is determined using the instance-specific SVM for the adaptive tracker, whereas the non-adaptive variation has to use the class-specific SVM.

6.3 Experiments

This section explores how non-adaptive and adaptive tracking influence the performance of finding heads compared to a head detector. To make this comparison possible, the same detection error tradeoff (DET) curves are used. A rather small dataset of two image sequences is used, which are described in sec. 4.5.6 in the head detection chapter. Sequence 1 is less challenging, as it shows just a single person in front of a simple background, whereas sequence 2 features three persons that frequently occlude each other.

6.3.1 Fixed parameters

Some parameters are kept fixed throughout the experiments. They are either not very interesting, or do not make much of a difference as long as they are within a reasonable range. Their values can be found in table 6.1.

The threshold of the SVM that is used to detect heads is chosen such that in combination with the initialization logic of the tracker there are only very few false positives. It is possible to suppress false positive track initiations altogether with a higher threshold, but the variance of the results gets fairly high, which makes them unreliable. On top of this, it may not be very realistic to not expect at least few wrong track initializations.

Detector and trackers use **FHOG6 4x8** as features. The class-specific measurement model uses the same classifier as the detector, which is trained on frontal heads only. Using another classifier that is trained on a broader class does not lead to very different results and is omitted for clarity.

The learn rate λ is rather high to allow for a quick adaptation. As opposed to the short-term tracker of the previous chapter, the template drift is kept in check with the class-specific model, which allows for the higher learn rate.

Parameter	Symbol	Value
Particle count	n	500
	σ_x	0.2
Motion model deviations	σ_y	0.2
	σ_s	0.05
Min window height [px]		40
Negative sample count per frame		10
Instance-specific SVM penalty multiplier	C	10
Learn rate	λ	0.5

Table 6.1: Parameters of the visual tracker that are kept constant throughout the experiments.

6.3.2 Methodology

The results include the DET curve of the head detector, evaluated on the same video, to enable the comparison between tracking and detection. Its performance is similar to a non-adaptive tracker whose detection and disappearance thresholds are kept equal to each other. A tracker on the other hand can have different thresholds to ensure a low false positive rate on initialization and a low miss rate once a target was initialized.

The first experiments are conducted without the adaptive component of the tracker, so the influence of having different score thresholds for initialization and termination can be seen. To generate the DET curve, the SVM threshold that determines the disappearance of a head is initially set to 1.0. For each subsequent experiment, the threshold is decreased by 1.0 to generate different points on the DET curve. Linear interpolation roughly determines the points in between.

The adaptive component is only used in the second experiment. Again, the SVM threshold that determines the disappearance is changed to generate the DET curve, but this time the score of the instance-specific SVM is used instead. Beginning again at 1.0, the threshold is subsequently decreased by 0.25 to generate different points on the curve.

The randomness of the particle filter introduces some variance into the results. Because of that, each experiment is repeated 25 times and the resulting number of false positives and misses is averaged.

6.3.3 Results

As can be seen from the DET curves in figures 6.1 and 6.2, making use of the continuous detection confidence by tracking heads clearly improves the performance and finds more heads at the same false positive rates. Adding an adaptive component to the tracker has less impact and in some cases even shows a worse performance.

This result might be a bit surprising, as the adaptive measurement model was thought to enable tracking of non-frontal heads in the first place. An explanation can be found in the training of the head classifier: Non-frontal heads were not included in the positive examples, but they were not included in the negatives, either. So potentially they result in higher scores than real negatives and enable the tracking of heads regardless of pose, given that the SVM threshold is low enough.

Figures 6.3 and 6.4 give some insight into this explanation. Figure 6.3a shows the frame with head detections (white bounding boxes) and tracked heads (colored bounding boxes). Figures 6.3b, 6.3c, and 6.3d show the class-specific, instance-specific, and combined probability density of one tracked head across the frame. Figure 6.4 is a close-up of the head in question, where the probability is scaled such that the maximum on the head is red and one hundredths of that is desaturated gray.

It can be observed that the profile head has a small class-specific probability (and hence a small SVM score), so that it is not detected. Nevertheless, there is still a peak at the head's center, which guides the tracker. This is especially visible in figure 6.4a.

Another observation is the distinction between head and background. The class-specific probability map shows high, thin peaks and large differences between high and low probabilities, whereas the instance-specific probability map is much smoother with one clear maximum and less difference between high and low probabilities. The reason for that is the much larger dataset that was used to train the class-specific SVM, which leads to a narrower margin between positive and negative class compared to the instance-specific SVM. This is depicted in figure 6.5, where the exemplary positive and negative patches (dark green and red) are within the margin for the instance-specific SVM, but well outside the margin for the class-specific SVM.

A higher distinction between high and low probabilities leads to more influence on the combined result, which means that the impact of the instance-specific

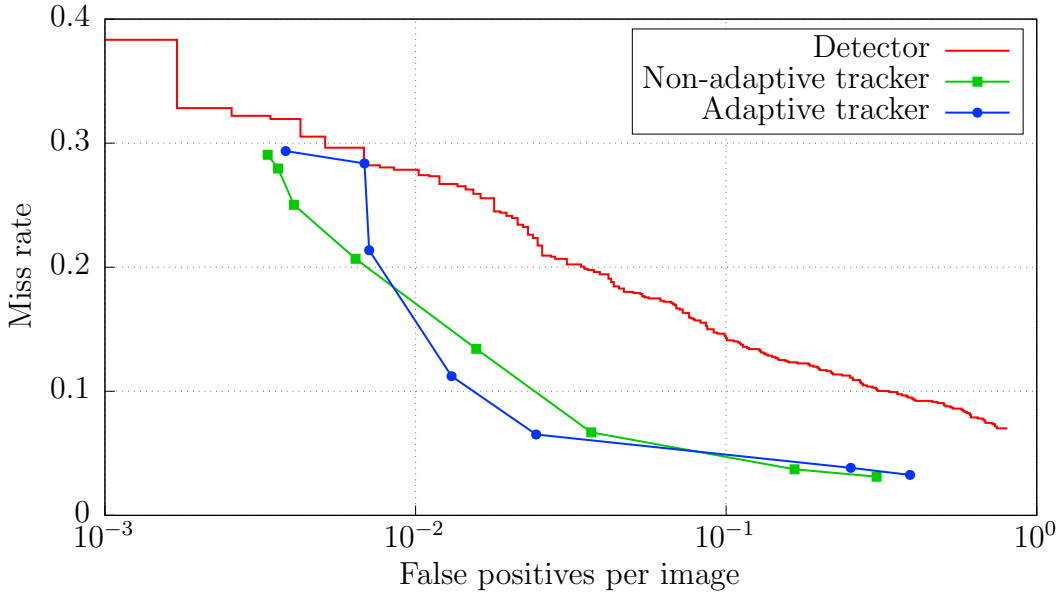


Figure 6.1: DET curves of head detector and trackers (using features **FHOG6 4x8**) tested on sequence 1.

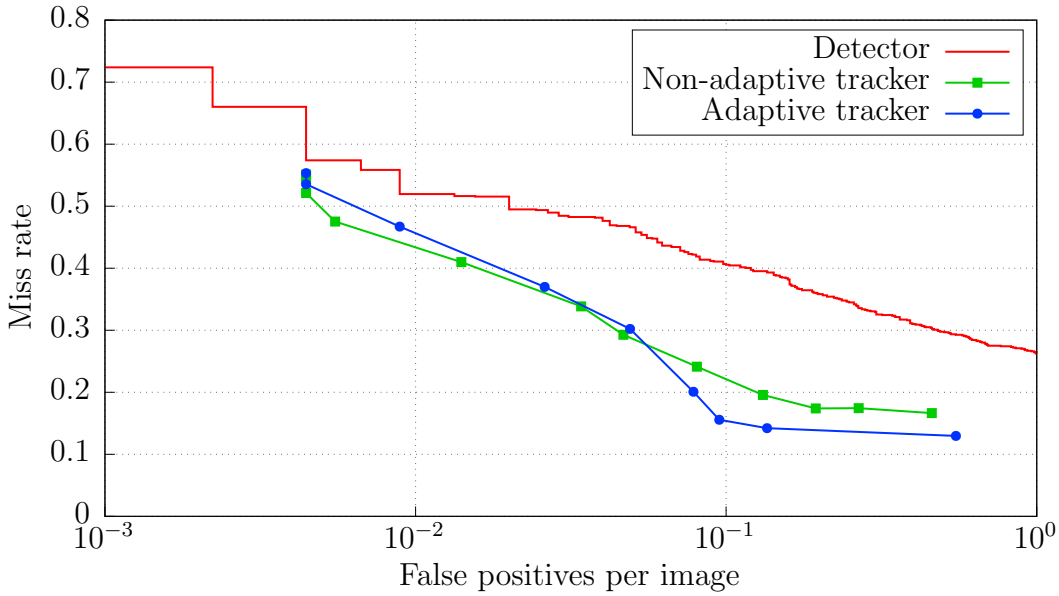
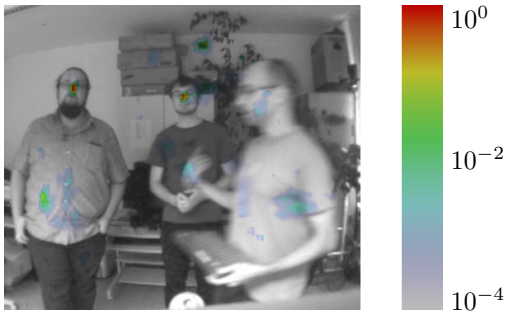


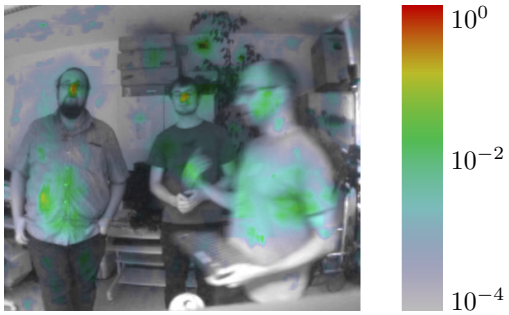
Figure 6.2: DET curves of head detector and trackers (using features **FHOG6 4x8**) tested on sequence 2.

(a) Detections and tracks



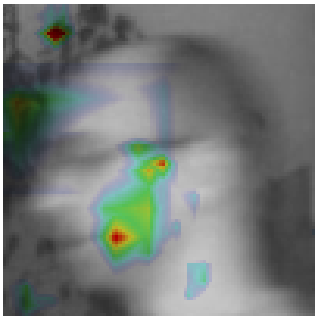
(b) Class-specific probability map

(c) Instance-specific probability map

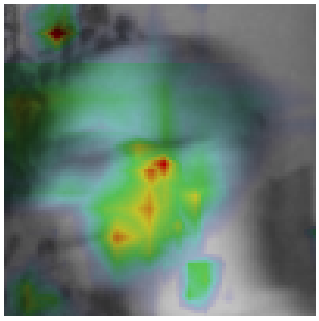


(d) Combined probability map

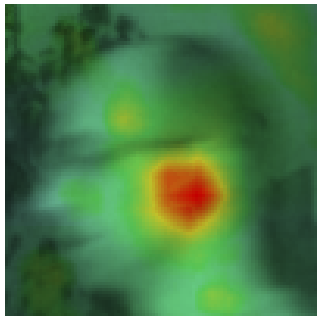
Figure 6.3: Detections (white bounding boxes), tracks (colored bounding boxes), and probability heat maps.



(a) Class-specific

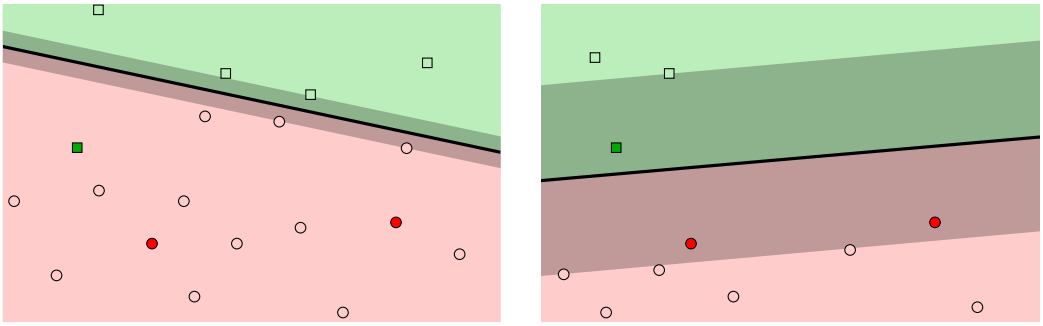


(b) Combined



(c) Instance-specific

Figure 6.4: Closer look at the probability heat maps.



(a) Class-specific SVM (b) Instance-specific SVM

Figure 6.5: Closer look at the support vector machines with training data (bright green and red) and patches of the current frame (darker green and red). The instance-specific SVM adapted to the actual image sequence and its margin is much wider due to less training examples.

probabilities is diminished. This can be observed in figure 6.4b, where the combined probability map shows the same peaks as the class-specific map. The instance-specific probabilities merely influenced their strength, as the peaks closer to the instance-specific probability peak are more pronounced. Thus, the adaptive component has only little influence on the actual target localization and tracking, which explains the similar results of both trackers.

With only little influence on the tracking itself, the adaptive component's main task is confirming the visibility, which is also its strength. As long as there is no sudden, drastic appearance change, the head will receive a high SVM score. Additionally, there are no similarly high peaks in the immediate surroundings, so that drift is easily detectable once the SVM score drops for the estimated head position. The choice of the termination threshold is thus more intuitive than for the non-adaptive tracker, which needs thresholds way below -1 for a similar performance, thereby increasing the risk of drifting to other nearby peaks.

6.4 Conclusion

For the tracking application, the SVM threshold is chosen such that there are very little false positives. Some peaks in the detector confidence density are therefore below the threshold and yield no detection. The head tracker can utilize these peaks to guide the particles as long as its SVM score is above the visibility threshold. This leads to a lower miss rate compared to the detector.

The results suggest that the adaptive model does not improve the performance by much. However, in order to compete, the non-adaptive tracker must use a very low score threshold to determine the head visibility. Because of that, choosing a proper value for the threshold is quite hard. This is much more intuitive for the adaptive tracker, which is its biggest advantage.

By utilizing the same features, the tracker can re-use the feature pyramid of the detector. Therefore, the most expensive computations are already done. The tracker itself does not add much on top and thus is only marginally slower than the detector alone.

As the proposed head tracker[1] works quite differently compared to the Kalman-filter-based people tracker, it is not simple to combine the two approaches. Hence, creating a new particle-filter-based people tracking algorithm instead might be the better choice.

[1]The source code can be found at https://github.com/ex-ratt/AdapTrack.

Chapter 7

People Tracking - Version 2

The previous chapter showed the advantages of using the continuous detection scores with a particle filter as opposed to working with the final detections only. To utilize this in a people tracking framework, it therefore has to be built on top of a particle filter. This chapter introduces the revised people tracker that tries to overcome some of the weaknesses of the first version (see chapter 3).

Figure 7.1 illustrates the steps of the revised people tracking algorithm. The sensor cues each prepare a measurement model, which stores detections as well as the original measurement. The model is responsible for data association, weighting the particles, determining visible tracks, and initiating new ones. Track removal and remaining filtering steps are taken care of by the tracker, as is the generation of the final output.

7.1 State and motion model

State as well as motion model remain unchanged from the first version. Thus, the state consists of the three-dimensional head position and a two-dimensional velocity vector, resulting in a state vector with five elements: $\mathbf{x}_k = (x_k, y_k, z_k, \dot{x}_k, \dot{y}_k)^T$. The probability density distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ of the state is approximated by a set of particles, as introduced in the previous chapter.

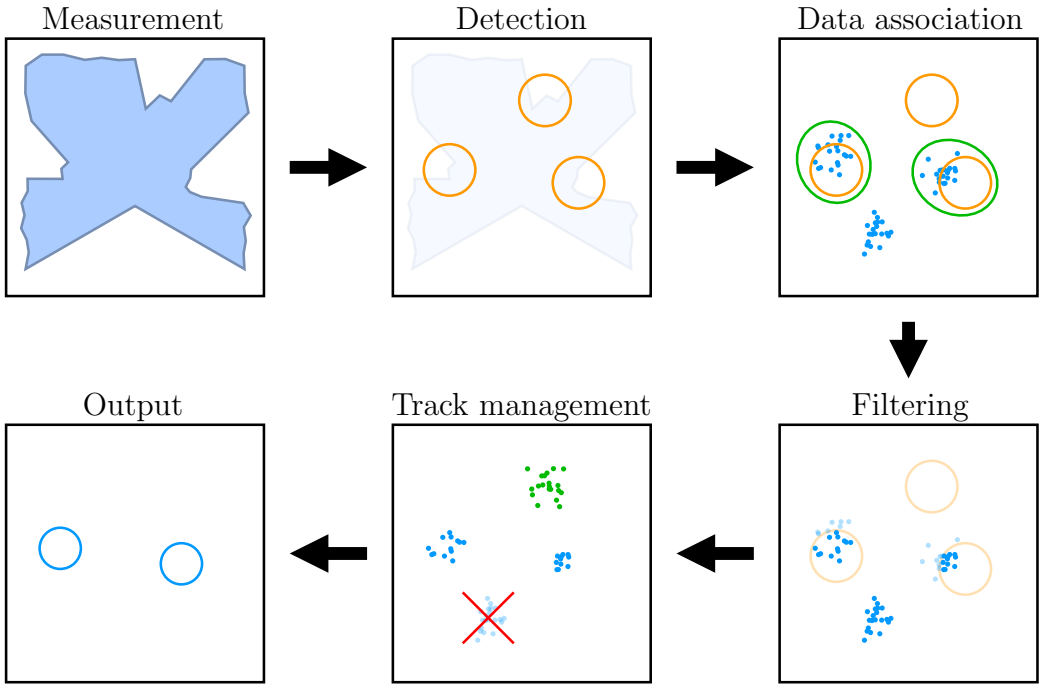


Figure 7.1: Steps of the revised people tracking algorithm. Orange circles represent detections, while blue and green dots are the particles of the tracks.

The motion model samples from the process noise to determine the new particle positions, the matrices F_k and Q_k are defined in equations 3.2 and 3.3, respectively. The particle index is dropped for clarity.

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(0, Q_k) \tag{7.1}$$

7.2 Measurement models

Each sensor cue s provides its own measurement model to update the particle's weights. They differ depending on the kind of measurement they are based on. The different models are assumed to be independent, so their probabilities can be multiplied to yield the particle's importance factor:

$$\pi_k^{[i]} \propto \prod_s p_s(\mathbf{z}_k | \mathbf{x}_k^{[i]}) \tag{7.2}$$

Usually though, the sensor cue's measurements arrive at different times, so an explicit multiplication is rarely needed. The only exception is the proximity model, as its measurement can be generated at any time.

7.2.1 Gaussian position estimates

The sensor cues of the first version of the people tracker provide three-dimensional Gaussian position estimates that are generated from the detections of persons within a sensor measurement. To support experiments with hybrids of the first and revised tracker, a measurement model that computes the particle's measurement probability based on these position estimates is needed.

First, for each track, mean and covariance matrix are computed from the particle's positions. The data association then is the same as in the first tracker, using the Mahalanobis distance between the track's and measurement's Gaussians. Only tracks with an assigned position estimate are updated. The measurement probability of a particle is computed from the probability density of the Gaussian, evaluated at the particle's position.

7.2.2 Laser range scan

The laser scan cue uses the same clustering approach as in the first people tracker with two differences. It does not cluster the measurements into groups of persons and it uses the existing tracks to assign the legs to instead of assigning the legs to each other to determine the positions of persons.

The laser range measurements are first transformed into points of the map coordinate system. This is followed by two filtering steps. Measurements of static obstacles are removed by background subtraction with a known occupancy grid map of the environment. The remaining measurements are segmented using single-linkage clustering with a minimum distance of 5 cm and segments with less than two points are discarded.

Then, all of the points that were not removed are assumed to be caused by human legs. A complete-linkage clustering with a maximum diameter of 25 cm segments the points into approximately leg-sized clusters. The center point marks the leg position.

Tracks and legs are then associated to one another, where each leg can be assigned to at most one track, but each track can have up to two legs. The association is based on the Euclidean distance in the ground plane with a maximum distance between track and leg of 50 cm.

A Gaussian distribution centered at the middle point between the legs is used to compute the measurement probability of the particles that belong to the track assigned to the legs.

7.2.3 Camera image

The camera image cue uses the measurement model introduced in the previous chapter. In contrast to the purely visual head tracker though, tracks are not terminated upon occlusion, as it may only be temporary and may not affect all the other sensors at the same time. In this case, the measurement probabilities (and thus, the particle weights) will drop very low, even though the hypothesized state might be correct. If there is another distracting object close by (e.g. another person's head), it will attract the particles far more than the real person.

This problem becomes more clear when considering the field of view. The previous chapter's head tracker simply assigns a weight of zero to particles outside the camera's field of view, because the space of possible states is restricted to the image. When tracking people in the real world with several sensors, particles must be allowed outside the observable area of the camera, just as they must be allowed to be occluded or otherwise imperceivable.

To this end, a binary variable v_k is introduced, which indicates whether a person is visible to the camera. If she is, then the probability can be computed from the measurement. Otherwise, the measurement could be anything and the probability is equally distributed, independent of the actual state. This is reflected by the two terms of the measurement probability (particle index is dropped for clarity):

$$p(\mathbf{z}_k|\mathbf{x}_k) = p(\mathbf{z}_k|\mathbf{x}_k \cap v_k)p(v_k) + p(\mathbf{z}_k|\mathbf{x}_k \cap \bar{v}_k)p(\bar{v}_k) \tag{7.3}$$

Visibility v_k and state $\mathbf{x}_k$ are assumed to be independent. $p(v_k)$ is the probability that the person is visible, $p(\bar{v}_k) = 1 - p(v_k)$ is the opposite, $p(\mathbf{z}_k|\mathbf{x}_k \cap v_k)$ is the measurement probability in case the person is visible, and

$p(\mathbf{z}_k|\mathbf{x}_k \cap \bar{v}_k)$ is the measurement probability in case the person is not visible, and hence is independent from the actual measurement. It is represented by a constant chosen for this sensor cue, as is the visibility probability. The visual head tracker's measurement probability introduced in the previous chapter can be seen as a special case with $p(v_k) = 1$.

If there is no adaptive component, then the measurement probability for visible persons is only determined by the class-specific model $p_c(\mathbf{z}_k|\mathbf{x}_k \cap v_k)$, which yields high probabilities for head-like regions:

$$p(\mathbf{z}_k|\mathbf{x}_k \cap v_k) = p_c(\mathbf{z}_k|\mathbf{x}_k \cap v_k) \tag{7.4}$$

With the addition of an adaptive component, it is also influenced by the instance-specific measurement model $p_i(\mathbf{z}_k|\mathbf{x}_k \cap v_k)$, which yields high probabilities for regions that look similar to the tracked head:

$$p(\mathbf{z}_k|\mathbf{x}_k \cap v_k) = p_c(\mathbf{z}_k|\mathbf{x}_k \cap v_k)^{0.5} p_i(\mathbf{z}_k|\mathbf{x}_k \cap v_k)^{0.5} \tag{7.5}$$

To prevent underestimating the uncertainty caused by correlation between the two models, they are weighted by 0.5. See sec. 6.2.2 for an explanation.

7.2.4 Proximity

The proximity model is an exception from the other models, as there is no real measurement involved. As the persons are tracked independently from one another, there is nothing that prevents two tracks from occupying the same space. This may happen when there are ambiguities in the leg association and, more importantly, with the camera cues, as the same image pixels may be used to compute the particle weight's of different tracks. The head tracker solved this issue by removing the track that was occluded, but for the person tracker, temporary occlusions should not lead to the removal of tracks.

To prevent two tracks from merging, an artificial measurement $\mathbf{z}_k$ is created, which contains the predicted states $\mathbf{z}_{j,k}$ of all the other confirmed tracks but the one that is updated. Particles close to other confirmed tracks receive a measurement probability of lower than one [Hess and Fern, 2009, Cielniak

et al., 2010], which approaches zero if the distance is too low. Particles further away from other tracks are not penalized.

$$p(\mathbf{z}_k|\mathbf{x}_k^{[i]}) = \prod_{\mathbf{z}_{j,k}\in\mathbf{z}_k} p(\mathbf{z}_{j,k}|\mathbf{x}_k^{[i]}) \tag{7.6}$$

$$p(\mathbf{z}_{j,k}|\mathbf{x}_k^{[i]}) = \begin{cases} 1 & \text{if } d(\mathbf{z}_{j,k}, \mathbf{x}_k^{[i]}) \geq d_{max} \\ \frac{d(\mathbf{z}_{j,k}, \mathbf{x}_k^{[i]}) - d_{min}}{d_{max} - d_{min}} & \text{if } d_{min} < d(\mathbf{z}_{j,k}, \mathbf{x}_k^{[i]}) < d_{max} \\ 0 & \text{if } d(\mathbf{z}_{j,k}, \mathbf{x}_k^{[i]}) \leq d_{min} \end{cases} \tag{7.7}$$

Function $d(\mathbf{z}_{j,k}, \mathbf{x}_k^{[i]})$ computes the two-dimensional Euclidean distance in the ground plane between the particle state $\mathbf{x}_k^{[i]}$ and track $\mathbf{z}_{j,k}$.

7.3 Track initiation and termination

The measurement models decide whether a new track is necessary and initiate it based on the existing tracks and current measurement if need be. Similarly to the head tracking approach of the previous chapter, a new track needs to be confirmed a second time before it is reported. The camera-image-based detection cues are the only ones that can initiate and confirm new tracks, as they have the lowest false positive rates among the cues.

With each update, a measurement model also has to determine whether a track is still visible. This is done only after the particles were weighted and the states extracted. If a track is not confirmed by any cue for one second, then it is terminated. This is a very similar behavior to the first version of the people tracker.

The laser range scanner cue considers persons as visible if their track has at least one associated leg and was already confirmed by another cue before. Therefore, this cue can keep tracks alive that are not visible in the camera images anymore, but is not able to spawn new tracks.

The camera-image-based model projects the tracks's positions onto the image plane to yield the expected bounding boxes of the person's heads. The head

detector provides another set of bounding boxes according to the image contents, which are then assigned to the tracks in a one-to-one fashion based on the overlap ratio. Tracks with associated detection are considered visible, as are tracks whose SVM score of the head bounding box exceeds a visibility threshold. It uses the instance-specific SVM in case an adaptive model is used, otherwise the general head SVM is used. If a track was already confirmed and is still considered visible, then its instance-specific SVM is adapted according to the current appearance. Detections that are not associated with a track are used to initiate new (unconfirmed) tracks.

Data association of sensor cues that provide Gaussian position estimates is based on the Mahalanobis distance. Tracks with assigned measurement are considered visible, while Gaussian measurements without an associated track are only used to initiate new tracks for the visual cues. To stay consistent with the new tracker, laser-based cues do not initiate new tracks.

7.4 Multi-sensor data fusion

Each sensor cue that provides measurements with an associated measurement model is registered at the tracking system. The cues are expected to regularly send new measurements to update the state estimations.

Out-of-sequence measurements are handled by a buffering strategy that orders the incoming measurements according to their timestamp. Using a buffer with a fixed length, e.g. of a few hundred milliseconds, is suboptimal in most cases: If a measurement lags behind by more than that, then it will be discarded, and if it is delayed by less, then the buffer introduces an unnecessary delay of the output. By choosing the buffer size to be dynamic and processing the oldest measurement as soon as every sensor cue added at least one measurement, the output delay will be just as high as necessary without discarding any measurement.

This naive dynamic buffering strategy can be improved upon, which further reduces the output delay. The timestamp of a measurement is known as soon as it arrives at the sensor cue. It can be announced to the tracker at the time of arrival, before applying a detection algorithm. If the tracker knows the timestamps of each cue's next measurement, then this enables it to process the oldest measurement in the buffer even if the following measurements are not fully prepared yet.

The improved dynamic buffering strategy works as follows. As soon as a measurement arrives at one of the sensor cues, it is announced to the tracking system. Then, the cue prepares the measurement, e.g. by computing features or detecting persons, and afterwards sends the prepared measurement to the system. The tracking system itself keeps track of all the announced and prepared measurements. It will process the oldest measurement if it is ready and every other cue has at least announced its next measurement. Thus, each measurement is used as soon as possible without risking to have to discard one that needs a bit longer to prepare.

7.5 Experiments

To analyze the influence of each component on tracking performance and efficiency, multiple experiments are conducted. In each experiment, single components are exchanged in order to determine whether and how they improve the tracking results.

Comparisons with state-of-the-art people trackers can be made in two ways: By using the dataset other trackers were evaluated with or by using their software to produces the results. Some people trackers are available through the Robot Operating System (ROS). Unfortunately, ROS only supports pinhole camera models, which prevents using omni-directional images. A comparison using the dataset created for this thesis is thus not possible. At the time of this writing, there are no other datasets available that feature mostly standing and moving persons, recorded by a camera in head height and a laser scanner in leg height. Therefore, a direct comparison with state-of-the-art trackers is not possible.

7.5.1 Dataset

A single annotated dataset is used in the experiments. It was recorded in a museum with six persons, simulating an interaction between visitors and robot. The robot stays next to an exhibit, while the persons are mostly facing the robot and sometimes walk around. Some of the more problematic difficulties of the dataset are: legs not being visible in the laser range scan, occlusions caused by other persons or constructions on the robot itself, and overexposure. The record has a length of 267 seconds.

The person's positions are annotated every second. They consist of the head position that is projected onto the ground plane. The laser scan was the most important guidance, as the image alone does not suffice to estimate the distance properly. In some cases though, there is no range reading from the laser scanner, and the position had to be estimated roughly based on the image and environment map of the museum. With range readings, the image is still useful in determining whether the head is above one of the legs or in between, or in some rare cases in front of the legs when a person is bent forward in order to have a better look at the robot's screen. The annotated positions therefore are only accurate up to few centimeters.

7.5.2 Evaluation methodology

The CLEAR-MOT metrics [Bernardin and Stiefelhagen, 2008] are used to evaluate and compare the tracking performances. Each annotated frame k consists of a number g_k of ground truth positions, which are assigned to the tracker output in a one-to-one manner. Three types of errors are counted for each frame: The number of misses m_k, false positives fp_k, and mismatches mm_k.

Misses are persons without a matching track, false positives are tracks that do not match with any ground truth position, and mismatches happen if one track switches from one person to another, but also when one person is lost and later re-detected, which results in a new track.

The multiple object tracking accuracy (MOTA) combines these three errors into one metric:

$$MOTA = 1 - \frac{\sum_k (m_k + fp_k + mm_k)}{\sum_k g_k} \tag{7.8}$$

A perfect tracker would have a MOTA score of 100 %, whereas a very bad tracker can achieve negative values. For further insights into the errors made by the trackers, individual error rates are given, which are computed by dividing the number of errors by the ground truth count.

A second metric evaluates how well the persons are localized using the distance d_k^i between each of the c_k matched pairs of frame k. This results in the multiple object tracking precision (MOTP) metric:

$$MOTP = \frac{\sum_{i,k} d_k^i}{\sum_k c_k} \tag{7.9}$$

Associations between ground truth and tracking output are only valid if their distance d_k^i does not exceed a threshold, which is set to 0.5 m throughout the experiments. This means that the MOTP score can never exceed this threshold and will probably not go over 0.25 m even for bad trackers.

For each annotated frame, the evaluation checks whether the previous matches are still valid by comparing their distance to the threshold. A global nearest neighbor algorithm associates ground truth and tracking output that was not matched the frame before or is not matched anymore because of the distance. Remaining annotations count as misses, remaining tracked persons count as false positives, and each previous match that is violated by a new association counts as mismatch. A detailed explanation of this algorithm can be found in [Bernardin and Stiefelhagen, 2008].

To make the results more reliable, the number of ground truth annotations, errors, matches, and match distances are added across five runs to yield an average score. For individual runs, the error rates and MOTA scores deviate by up to one percent point from this average, while the MOTP score rarely deviates by more than 2 mm.

7.5.3 Components

The tested trackers are combinations of components, where a component is either the central tracking algorithm or one of its sensor cues. To determine the influence of each component, different configurations are compared to each other, where one component is exchanged at a time. The components are:

T1: First people tracker version based on the Kalman filter.
T2: Revised people tracker based on a particle filter.
L1: Laser scan cue of the first tracker, which assigns detected legs to one another in a pairwise manner to determine the position of persons.

Configuration	MOTP	MOTA	Miss rate	FP rate	MM rate
T1–L1	60 mm	48.6 %	32.5 %	17.7 %	1.3 %
T2–L1	63 mm	48.7 %	32.8 %	17.2 %	1.2 %
T2–L2	64 mm	47.0 %	32.8 %	18.9 %	1.4 %

Table 7.1: Comparison of the laser scan cues.

L2: Laser scan cue of the revised tracker, which assigns the detected legs directly to tracks.
F: Frontal face detection cue of the first tracker, which results in Gaussian head hypotheses in world coordinates.
U: Upper body detection cue of the first tracker, which results in Gaussian head hypotheses in world coordinates.
H1: Head detection cue of the first tracker, which results in Gaussian head hypotheses in world coordinates.
H2: Non-adaptive head detection cue of the revised tracker, where particles are weighted according to the SVM score of their associated bounding box in the image.
H2A: Adaptive head detection cue of the revised tracker, where particles are weighted according to the SVM scores of their associated bounding box in the image.
P: Proximity model that decreases the weights of particles that are close to another confirmed track.

7.5.4 Results

The **first experiment** compares the laser scan cues of the trackers, see table 7.1 for results. The parameters and cues had to be changed in order to provide tracking output, as otherwise visual detections would be necessary to confirm the tracked persons. **L2** initiates new tracks similar to **L1** by assigning the remaining legs in a pairwise manner and using their centers to sample particles. Both cues need two observations of a person within one second to confirm her existence. An exception is the front laser scanner for **T1-L1**, as it provides twice as much scans compared to the rear laser scanner, and needs three detections. This is negligible though and is not reflected by the results.

Most of the false positives are caused by a static obstacle that is not registered in the map and results in a track that exists for almost all the time. This is

the reason why usually, detections by the laser scan cue alone should not be sufficient to confirm a person's existence, as the background subtraction and clustering approach classifies anything as a human leg that is not present in the map.

The performance of both trackers with **L1** is very similar. The main difference is the number of laser scans that is processed: The first tracker computes Gaussian person positions, asynchronously sends them to the tracker and immediately processes the next laser scan, while the second tracker needs the cues to wait until the particle weights are computed. Furthermore, the processing can be delayed slightly because of the buffering. Therefore, the configuration **T2-L1** processes only around 80 to 90 % of the laser range scans compared to **T1-L1**, which results in slightly less precision. Still, the performance is very close.

T2-L2 on the other hand shows a higher false positive rate. In some rare situations, the laser range readings of a person result in three detected legs instead of only two, resulting in a single false positive detection. For trackers with the **L1** cue, the initiated track is not confirmed and discarded quickly, provided that there is only one false detection within a second. **L2** on the other hand does assign detected legs directly to the closest track, which in this case confirms the false track, as it takes over one of the legs from the correct track. One of both tracks is terminated only when just one leg is visible.

The **second experiment** compares the visual detection cues, see table 7.2 for results. The chosen parameters are aimed at a low false positive rate, which can be seen by comparing the values to the laser cues. The reasoning is as follows. The robot can only look at one person at a time, so an outsider cannot see how many people the robot actually perceives at a given time. But if it does not look at a person, and instead keeps staring into random directions, then this can deteriorate the impression that the robot perceives persons in its vicinity. Therefore, the SVM threshold for detecting heads is set to 1. The parameters of the OpenCV Viola-Jones detectors are chosen similarly.

The performance of the tracking declines when exchanging face and upper body detection by head detection (**T1-F-U** vs. **T1-H1**). The former two cues complement each other well, which is lost when only using a single detection cue, and results in a higher miss rate and less precision.

When replacing the underlying tracker, precision and number of mismatches increase. This is caused by the limited number of particles. If there is a bigger

Configuration	MOTP	MOTA	Miss rate	FP rate	MM rate
T1–F–U	127 mm	48.1 %	38.8 %	6.4 %	6.7 %
T1–H1	193 mm	40.7 %	47.2 %	4.5 %	7.6 %
T2–H1	171 mm	40.2 %	47.2 %	4.2 %	8.5 %
T2–H2	159 mm	48.2 %	42.3 %	4.3 %	5.2 %
T2–H2–P	155 mm	51.2 %	42.2 %	1.4 %	5.3 %
T2–H2A–P	156 mm	59.5 %	36.1 %	1.6 %	2.0 %

Table 7.2: Comparison of the visual cues.

discrepancy between track and detection, e.g. caused by sudden movement or missing detections beforehand, then the Kalman filter corrects the estimate much more quickly.

The correction capability of the particle filter on the other hand is restricted by the motion model if there are no particles in the right spot, as they have to increase their velocity over time until some of them are where they need to be. Increasing the particle count leads to a quicker reaction, as few particles are spread further, but it is still limited. In some cases, the slow adaptation leads to the detection in the next frame being already too far away to be associated with the track anymore. Then, a new track is initiated, which leads to a mismatch.

On the other hand, the very quick adaptation of the Kalman filter leads to wrong associations in some cases, where two persons are close to one another and one is detected, while the other is not. If the situation changes and the other person is detected, then the Kalman filter will quickly switch over, while the particle filter cannot react as fast to this outlier measurement. If the persons are closer than 0.5 m, then the evaluation will not count it as a mismatch, and the high distance between track and associated ground truth annotation increases the MOTP score of **T1-H1** compared to **T2-H1**.

T2-H2 performs better than **T2-H1** due to lower miss and mismatch rates. It is caused by **H2** using the detector confidence, which allows it to follow the head of a person even if its SVM score drops below 1.0. The addition of the proximity model **P** reduces the number of false positives, which are often caused by one person occluding another, which attracts the occluded track to the foreground person. Without the proximity model preventing this, the non-occluded person might be tracked twice. Compared to **H2**, the adaptive variation **H2A** can follow heads even longer, which leads to further reduced miss and mismatch rates.

Configuration		MOTP	MOTA	Miss rate	FP rate	MM rate
T2–H2–P	-1.0	155 mm	51.2 %	42.2 %	1.4 %	5.3 %
T2–H2–P	-3.0	158 mm	58.2 %	36.5 %	2.3 %	3.0 %
T2–H2–P	-5.0	157 mm	58.2 %	35.0 %	4.1 %	2.8 %
T2–H2–P	-7.0	162 mm	57.3 %	34.5 %	5.3 %	3.0 %
T2–H2A–P	0.00	156 mm	59.5 %	36.1 %	1.6 %	2.0 %
T2–H2A–P	-0.25	158 mm	59.5 %	33.8 %	4.0 %	2.7 %
T2–H2A–P	-0.50	160 mm	62.0 %	32.4 %	3.2 %	2.4 %
T2–H2A–P	-0.75	165 mm	55.7 %	31.2 %	10.3 %	2.9 %

Table 7.3: Comparison of SVM score visibility thresholds of non-adaptive and adaptive visual cues.

For both **H2** and **H2A**, the SVM threshold that determines the visibility of a person is chosen rather conservative to keep the false positive rate low. **H2** uses a threshold of -1.0, while **H2A** uses a threshold of 0.0. In both cases, lower thresholds would lead to a better performance on the chosen benchmark sequence, but this may change in other scenarios. Table 7.3 shows the results of the **third experiment**, which compares performances for different thresholds.

The performance of the non-adaptive tracker increases fairly well with a lower threshold, whereas the adaptive tracker does not improve as much. At their best, the difference is not quite as big, which was already discussed in the previous chapter.

The **fourth experiment** combines visual with laser-based cues and analyzes how the individual strengths translate to a tracker with more components. Table 7.4 shows the results. There are differences compared to the isolated experiments, as some of the shortcomings can be compensated by other sensor cues.

One notable difference is the exchange of the detection cues **F-U** in favor of head detection **H1**. The reduction in precision is not as severe and the miss rate decreases considerably. This is because the laser scan can confirm most of the persons and keep their tracks alive, which was not the case in the visual-only experiments. There are a few cases of persons that are not perceived by the rear laser scanner, although the persons are close enough, because their pants do not reflect the laser light well. As opposed to visual-only tracking, the face and upper body detector only search near existing

Configuration	MOTP	MOTA	Miss rate	FP rate	MM rate
T1–L1–F–U	79 mm	66.2 %	28.3 %	2.9 %	2.6 %
T1–L1–H1	95 mm	73.1 %	20.0 %	3.2 %	3.7 %
T2–L1–H1	101 mm	75.1 %	17.6 %	3.2 %	4.1 %
T2–L1–H2	117 mm	73.7 %	17.7 %	6.0 %	2.6 %
T2–L1–H2–P	109 mm	79.9 %	16.0 %	2.0 %	2.2 %
T2–L2–H2–P	102 mm	81.2 %	15.3 %	1.4 %	2.1 %
T2–L2–H2A–P	98 mm	81.7 %	14.7 %	1.9 %	1.8 %

Table 7.4: Comparison of trackers with both laser and visual cues.

tracks and thus rely on the laser scan cue. The head detector on the other hand is more efficient and searches the whole image, resulting in the person being detected and the performance increased.

Other improvements are less pronounced compared to the individual tests, because other cues compensate the errors made and thus there is less potential for further improvement. This is especially noticeable with the adaptive head tracking **H2A**, which has only little influence now.

Compared to laser-only tracking, persons that are only detected by the laser cue neither initiate new tracks nor confirm them, which suppresses many of the false positives. Hence, the disadvantage of **L2** compared to **L1** does not come into effect.

Overall, the revised people tracker **T2-L2-H2A-P** has about half the miss rate of the first version **T1-L1-F-U**, while also reducing false positives and mismatches. The location of the tracked persons is not as precise though. This is due to the fact that most of the former misses are of persons that are not visible in the laser scan and thus have to rely on the localization capabilities of the visual cues, which are not as precise in estimating the distance from the robot. Figure 7.2 shows an example situation with a person in profile view and other persons that are not perceived by the laser scanner.

Computation time and update rates are measured in the **fifth experiment**. Table 7.5 shows the average computation time of the tracker and each of its cues, the number of updates per second that actually occurred, and the average delay of the tracking output. In case there were two cues of the same type, the slower time of both is presented.

The image cues take much more time than the laser scan cues, which is hardly surprising given the amount of data and complexity. The first tracker

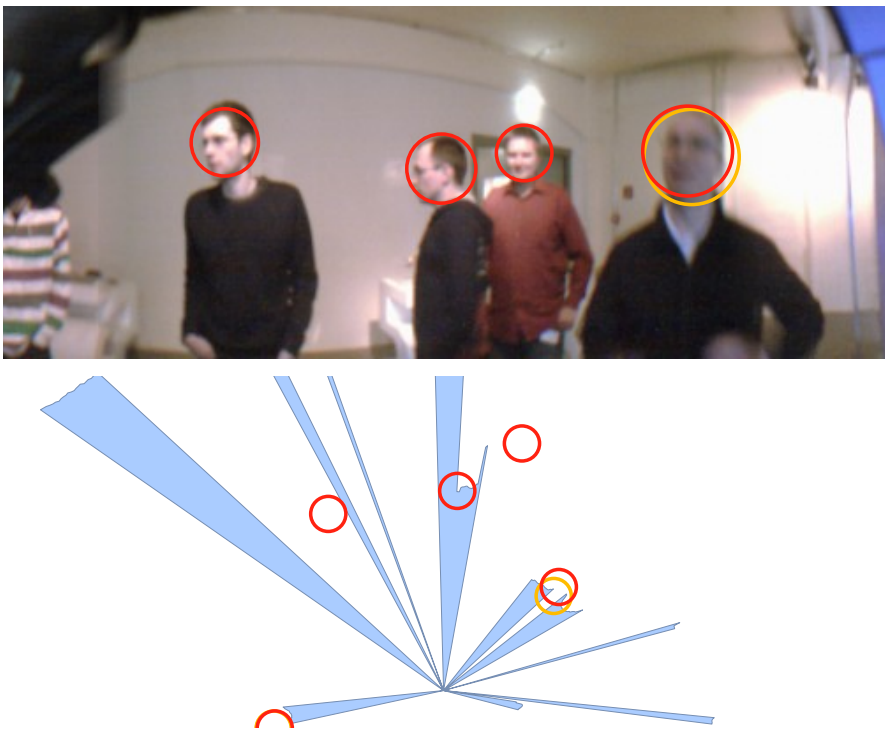


Figure 7.2: Clipped omni-directional image and top-down-view of laser scan with tracking outputs of **T1-L1-F-U** (yellow) and **T2-L2-H2A-P** (red).

Configuration	Laser	Image	Tracker	Rate	Delay
T1–F–U	-	208 ms	< 1 ms	7.7 Hz	187 ms
T1–H1	-	69 ms	< 1 ms	8.8 Hz	78 ms
T1–L1–F–U	7 ms	101 ms	< 1 ms	23.4 Hz	57 ms
T1–L1–H1	2 ms	69 ms	< 1 ms	23.6 Hz	60 ms
T2–H2–P	-	77 ms	2 ms	8.9 Hz	125 ms
T2–H2A–P	-	83 ms	2 ms	8.9 Hz	133 ms
T2–L2–H2–P	6 ms	85 ms	4 ms	20.4 Hz	186 ms
T2–L2–H2A–P	6 ms	92 ms	4 ms	20.1 Hz	192 ms

Table 7.5: Comparison of computation times, update rates and delay of different trackers.

version is faster than the revised tracker, despite reprocessing the data in case of out-of-sequence measurements. Additionally, the data association is outsourced into the cues for the revised tracker and therefore does not count into its computation time. It does contain the proximity measurement model however.

The head image cue is a bit slower for the revised tracker, as it has to compute the data association and particle weights in addition to detecting heads. Adaptivity does not add very much on top of this. This depends on the number of persons of course, but the same applies to data association and tracker update.

The old visual-only tracker needs twice as much time for its image cues compared to the variation with laser cues. This is because without the laser cue's detections, the whole image has to be searched for the presence of persons, whereas with other detections, the search space is restricted. The head detection cue on the other hand is fairly fast and always examines the whole image, so there is no difference in time.

The update rate is limited by the number of sensor messages per second, which is 9 Hz for the camera images, 12 Hz for the front laser scanner, and 5 Hz for the rear laser scanner. Thus, the maximum update rate possible would be around 25 Hz. The head-cue-based visual-only trackers process almost every image, while the face- and upper-body-cues each process about half of them. When integrating the laser cues, the update rate is lower for the revised tracker, as the buffering causes the sensor cues to wait, which in some cases leads to sensor messages being missed.

When using a buffering strategy to cope with out-of-sequence measurements, the delay between sensor measurement and tracking output is affected, as the measurements can only be processed if every cue has received a measurement. Buffering clearly increases the delay compared to reprocessing. The only exception is the old visual-only tracker, as the detections take longer and hence induce the delay. Adding laser cues reduces the delay for the old tracker, as their detection is pretty fast, while the new tracker has an increased delay, as there is more potential for other cues to wait. A delay of 200 ms is acceptable, but these values come from reading sensor messages from a recorded file, rather than from a robot in live operation. If the hardware read-out of a sensor takes much more time, then the delay will suffer with a buffering strategy or else measurements must be discarded.

The naive version of the dynamic buffering strategy results in a delay of 252 ms (not shown in the table), so the improved version reduces the output delay by almost 25 % in this experiment.

7.6 Conclusion

By building on the head tracker established in the previous chapter, a new and improved people tracker is created. The only potential downside is the buffering strategy, which restricts the update rate and introduces a delay.

Both trackers suffer when there are many non-human obstacles in the laser scan that are not removed by the background subtraction. Examples are coats, bags, or changes of the environment that are not yet reflected by the map. Such objects lead to an over-estimation of the number of legs, which is the main reason why the laser scanner alone cannot confirm a track to be a real human. However, it can keep a track alive indefinitely once it was confirmed by a visual cue. In case of missing visual detections, such a track may drift away towards false positives in the laser scan.

The old tracker is more resistant to this kind of error: It immediately starts tracking false positives of the laser scan, even though it does not confirm them to be human. As they are already tracked, there is less of a chance that another track is associated to those false positives.

Chapter 8

Conclusion

This thesis explored ways to detect and track persons in the vicinity of a robot. This enables it to look at persons, which makes the interaction more natural. A first version of a people tracker was based on the state of the art and meant to be simple, so the most pressing issues and weaknesses could be identified. While the very simple data association for example did not pose a problem, the inability to visually detect persons viewed from the side was identified as one of the main issues.

To mitigate this weakness, a head detector was developed. It is quite fast compared to the previously used face and upper body detectors, but a high detection rate of heads in any orientation could not be achieved while also keeping the false positive rate reasonably low. The detector works well on frontal, upright heads, though. Because heads in other orientations were not included in the training set of the frontal head detector, their classifier scores usually are higher than for non-heads. This is exploited by the head tracker that makes use of the continuous classification scores across the image, which enables tracking heads regardless of pose after it was initialized by a frontal view. An adaptive component was ought to further improve the tracking performance, but has only a minor influence when choosing questionable parameters for the non-adaptive tracker.

Because the head tracker is based on a particle filter rather than Kalman filter, a new people tracker had to be designed around it. This also allowed for a slight change of the laser cue to make it more robust against crowded groups of persons. Compared to the old tracker, the miss rate could be cut in half, while also reducing false positives and mismatches. The adaptive

component has a positive effect, though it diminishes when combined with other sensor cues. Potential weaknesses are the delay that is introduced by the buffering strategy as well as false positives of the laser scan cue, as this has not been worked on.

The new people tracker is efficient and uses less computational resources than the old tracker, leaving more room for other important algorithms like navigation. It does not have special requirements like a dedicated GPU, cloud computing, or low latency communication, which makes it suitable for a wider variety of robots and operating conditions.

8.1 Further work

An area this work has not explored much are different motion models. While a near constant velocity model is a reasonable choice, assuming each motion to be caused by the intention of moving continuously into a direction can lead to undesirable side effects. Shifting the weight from one leg to another changes the head position, which is enough for the motion model to assume a constant movement into one direction, and leads to wrong predictions. If combined with a missed detection afterwards, the track might even drift away at high speeds, which could be observed in some of the experiments. A combination of constant velocity and random walk, where a small positional change does not influence the velocity as much, may already solve this problem. Several different motion models like random walk, constant velocity, and coordinated turn can be combined with an interacting multiple models filter [Linder et al., 2015]. More sophisticated motion models can further improve the prediction [Luber et al., 2010].

The laser scan cue is the biggest remaining issue, as it needs a map of the environment and an accurate localization, and may still produce lots of false positives. A classification-based approach [Weinrich et al., 2014, Leigh et al., 2015] would improve the performance of person detection in laser scans. Explicitly tracking legs and associating them to persons [Arras et al., 2008, Song et al., 2010] can make laser-scan-based tracking even more robust, as wrong associations between legs and persons are less likely compared to the proposed approach.

The laser scanner perceives any obstacle, which may or may not be a human, while the detectors applied to the camera images specifically find human

heads. This difference in perception is accounted for by the different reliability values in the first tracker version and by the inability of the laser-based-cues to spawn new tracks in the revised tracker. Explicit modeling of these perception capabilities [Munz et al., 2010] might improve the decision whether a person (still) exists or not.

Most of the processing time is spent on computing features and classifier responses in the image. The proposed detector searches heads in all areas of the image, even when the probability of existence is very low, e.g. directly above the ground or three meters above it. The processing time can be reduced when restricting the regions of interest inside the image based on a ground plane assumption [Sudowe and Leibe, 2011]. This would also decrease the delay induced by the buffering strategy, as the detection is the most time-consuming operation.

Similar to the first people tracker, an upper body detector might complement the head detector quite well and enable the robot to perceive persons further away. By re-using the same features as the head detector, the additional computations would take less time than a completely independent detector.

Finally, not explicitly handling occlusions in the head detection cue of the revised people tracker may be a mistake. In case one person occludes another, the same image information is used twice to correct two independent filters, which usually should be avoided [MacCormick and Blake, 1999]. Provisional experiments lead to a decreased performance, because small movements were often enough to cause the motion model to predict one person moving behind another. The track that incorrectly was assumed to be occluded could then not longer be updated. With a better motion model however, explicit occlusion handling may improve the tracking performance.

Bibliography

[Adam et al., 2006] Adam, A., Rivlin, E., and Shimshoni, I. (2006). Robust fragments-based tracking using the integral histogram. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 798–805.

[Andriluka et al., 2008] Andriluka, M., Roth, S., and Schiele, B. (2008). People-tracking-by-detection and people-detection-by-tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8.

[Arras et al., 2008] Arras, K. O., Grzonka, S., Luber, M., and Burgard, W. (2008). Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In *2008 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1710–1715.

[Arras et al., 2007] Arras, K. O., Martínez Mozos, Ó., and Burgard, W. (2007). Using boosted features for the detection of people in 2D range data. In *Proceedings 2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3402–3407.

[Babenko et al., 2008] Babenko, B., Dollár, P., Tu, Z., and Belongie, S. (2008). Simultaneous learning and alignment: Multi-instance and multi-pose learning. In *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*.

[Babenko et al., 2011] Babenko, B., Yang, M.-H., and Belongie, S. (2011). Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(8):1619–1632.

[Bar-Shalom et al., 2004] Bar-Shalom, Y., Chen, H., and Mallick, M. (2004). One-step solution for the multistep out-of-sequence-measurement problem

in tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 40(1):27–37.

[Bellotto and Hu, 2006] Bellotto, N. and Hu, H. (2006). Vision and laser data fusion for tracking people with a mobile robot. In *2006 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 7–12.

[Bellotto and Hu, 2009] Bellotto, N. and Hu, H. (2009). Multisensor-based human detection and tracking for mobile service robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):167–181.

[Bellotto and Hu, 2010] Bellotto, N. and Hu, H. (2010). Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of Bayesian filters. *Autonomous Robots*, 28(4):425–438.

[Benenson et al., 2014] Benenson, R., Omran, M., Hosang, J., and Schiele, B. (2014). Ten years of pedestrian detection, what have we learned? In *Computer Vision – ECCV 2014 Workshops: Proceedings, Part II*, volume 8926 of *Lecture Notes in Computer Science*, pages 613–627.

[Benfold and Reid, 2011] Benfold, B. and Reid, I. (2011). Stable multi-target tracking in real-time surveillance video. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3457–3464.

[Bernardin and Stiefelhagen, 2008] Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: The CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):1–10.

[Birchfield, 1998] Birchfield, S. (1998). Elliptical head tracking using intensity gradients and color histograms. In *1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 232–237.

[Blanco et al., 2003] Blanco, J., Burgard, W., Sanz, R., and Fernandez, J. (2003). Fast face detection for mobile robots by integrating laser range data with vision. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, volume 2, pages 953–958.

[Blaschko and Lampert, 2008] Blaschko, M. B. and Lampert, C. H. (2008). Learning to localize objects with structured output regression. In *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision, Proceedings, Part I*, volume 5302 of *Lecture Notes in Computer Science*, pages 2–15.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

[Bolme et al., 2010] Bolme, D. S., Beveridge, J. R., Draper, B. A., and Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2544–2550.

[Bouaynaya and Schonfeld, 2005] Bouaynaya, N. and Schonfeld, D. (2005). Complete system for head tracking using motion-based particle filter and randomly perturbed active contour. In *Proceedings of SPIE Image and Video Communications and Processing 2005*, volume 5685, pages 864–873.

[Brasnett et al., 2007] Brasnett, P., Mihaylova, L., Bull, D., and Canagarajah, N. (2007). Sequential Monte Carlo tracking by fusing multiple cues in video sequences. *Image and Vision Computing*, 25(8):1217–1227.

[Breitenstein et al., 2011] Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., and Van Gool, L. (2011). Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(9):1820–1833.

[Burgard et al., 1999] Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial intelligence*, 114(1-2):3–55.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–39. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Cielniak et al., 2010] Cielniak, G., Duckett, T., and Lilienthal, A. J. (2010). Data association and occlusion handling for vision-based people tracking by mobile robots. *Robotics and Autonomous Systems*, 58(5):435–443.

[Cox, 1993] Cox, I. J. (1993). A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66.

[Cox and Hingorani, 1996] Cox, I. J. and Hingorani, S. L. (1996). An efficient implementation of Reid's multiple hypothesis tracking algorithm and its

evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(2):138–150.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893.

[Dalal et al., 2006] Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Proceedings, Part II*, volume 3952 of *Lecture Notes in Computer Science*, pages 428–441.

[Dollár et al., 2014] Dollár, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(8):1532–1545.

[Dollár et al., 2010] Dollár, P., Belongie, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *British Machine Vision Conference (BMVC)*, pages 68.1–68.11.

[Dollár et al., 2009] Dollár, P., Tu, Z., Perona, P., and Belongie, S. (2009). Integral channel features. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–11.

[Dollár et al., 2012] Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(4):743–761.

[Doucet and Johansen, 2009] Doucet, A. and Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12:656–704.

[Elfes, 1987] Elfes, A. (1987). Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249–265.

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338.

[Felzenszwalb et al., 2010] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(9):1627–1645.

[Fortmann et al., 1983] Fortmann, T. E., Bar-Shalom, Y., and Scheffe, M. (1983). Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173–184.

[Gordon et al., 1993] Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113.

[Grabner et al., 2008] Grabner, H., Leistner, C., and Bischof, H. (2008). Semi-supervised on-line boosting for robust tracking. In *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision, Proceedings, Part I*, volume 5302 of *Lecture Notes in Computer Science*, pages 234–247.

[Groß et al., 2016] Groß, H.-M., Scheidig, A., Debes, K., Einhorn, E., Eisenbach, M., Mueller, S., Schmiedel, T., Trinh, T. Q., Weinrich, C., Wengefeld, T., Bley, A., and Martin, C. (2016). ROREAS: robot coach for walking and orientation training in clinical post-stroke rehabilitation–prototype implementation and evaluation in field trials. *Autonomous Robots*, 41(3):679–698.

[Grossberg, 1987] Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63.

[Hare et al., 2011] Hare, S., Saffari, A., and Torr, P. H. S. (2011). Struck: Structured output tracking with kernels. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 263–270.

[He et al., 2013] He, S., Yang, Q., Lau, R. W., Wang, J., and Yang, M.-H. (2013). Visual tracking via locality sensitive histograms. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2427–2434.

[Henriques et al., 2015] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. (2015). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 37(3):583–596.

[Hess and Fern, 2009] Hess, R. and Fern, A. (2009). Discriminatively trained particle filters for complex multi-object tracking. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 240–247.

[Hua et al., 2014] Hua, Y., Alahari, K., and Schmid, C. (2014). Occlusion and motion reasoning for long-term tracking. In *Computer Vision – ECCV 2014: 13th European Conference, Proceedings, Part VI*, volume 8694 of *Lecture Notes in Computer Science*, pages 172–187.

[Huang et al., 2005] Huang, C., Ai, H., Li, Y., and Lao, S. (2005). Vector boosting for rotation invariant multi-view face detection. In *Tenth IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 446–453.

[Isard and Blake, 1998] Isard, M. and Blake, A. (1998). Condensation–conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28.

[Ito et al., 2004] Ito, A., Hayakawa, S., and Terada, T. (2004). Why robots need body for mind communication - an attempt of eye-contact between human and robot. In *13th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN)*, pages 473–478.

[Julier and Uhlmann, 1997a] Julier, S. J. and Uhlmann, J. K. (1997a). A new extension of the Kalman filter to nonlinear systems. In *Proceedings of SPIE AeroSense Symposium*, volume 3068, pages 182–193.

[Julier and Uhlmann, 1997b] Julier, S. J. and Uhlmann, J. K. (1997b). A non-divergent estimation algorithm in the presence of unknown correlations. In *Proceedings of the American Control Conference*, volume 4, pages 2369–2373.

[Kalal et al., 2012] Kalal, Z., Mikolajczyk, K., and Matas, J. (2012). Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(7):1409–1422.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

[Khaleghi et al., 2013] Khaleghi, B., Khamis, A., Karray, F. O., and Razavi, S. N. (2013). Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28–44.

[Khan et al., 2004] Khan, Z., Balch, T., and Dellaert, F. (2004). A Rao-Blackwellized particle filter for EigenTracking. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 980–986. IEEE.

[Klein and Cremers, 2011] Klein, D. A. and Cremers, A. B. (2011). Boosting scalable gradient features for adaptive real-time tracking. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4411–4416.

[Klein et al., 2010] Klein, D. A., Schulz, D., Frintrop, S., and Cremers, A. B. (2010). Adaptive real-time video-tracking for arbitrary objects. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 772–777.

[Koditschek, 1987] Koditschek, D. (1987). Exact robot navigation by means of potential functions: Some topological considerations. In *1987 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 1–6.

[Kolarow et al., 2012] Kolarow, A., Brauckmann, M., Eisenbach, M., Schenk, K., Einhorn, E., Debes, K., and Gross, H.-M. (2012). Vision-based hyper-real-time object tracker for robotic applications. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2108–2115.

[Kristan et al., 2016a] Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Čehovin, L., Vojíř, T., Häger, G., Lukežič, A., and Fernández, G. (2016a). The Visual Object Tracking VOT2016 challenge results. In *Computer Vision – ECCV 2016 Workshops: Proceedings, Part II*, volume 9914 of *Lecture Notes in Computer Science*, pages 777–823.

[Kristan et al., 2015] Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Čehovin, L., Fernández, G., Vojíř, T., Häger, G., Nebehay, G., and Pflugfelder, R. (2015). The Visual Object Tracking VOT2015 challenge results. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 564–586.

[Kristan et al., 2016b] Kristan, M., Matas, J., Leonardis, A., Vojíř, T., Pflugfelder, R., Fernández, G., Nebehay, G., Porikli, F., and Čehovin, L. (2016b). A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(11):2137–2155.

[Kruppa et al., 2003] Kruppa, H., Castrillón Santana, M., and Schiele, B. (2003). Fast and robust face finding via local context. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 157–164.

[Kwon and Lee, 2010] Kwon, J. and Lee, K. M. (2010). Visual tracking decomposition. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1269–1276.

[La Cascia et al., 2000] La Cascia, M., Sclaroff, S., and Athitsos, V. (2000). Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3D models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(4):322–336.

[Lampert et al., 2009] Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2009). Efficient subwindow search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(12):2129–2142.

[Lau et al., 2010] Lau, B., Arras, K. O., and Burgard, W. (2010). Multi-model hypothesis group tracking and group size estimation. *International Journal of Social Robotics*, 2(1):19–30.

[Lebeda et al., 2014] Lebeda, K., Hadfield, S., and Bowden, R. (2014). 2D or not 2D: Bridging the gap between tracking and structure from motion. In *Computer Vision – ACCV 2014*, volume 9006 of *Lecture Notes in Computer Science*.

[Leibe et al., 2005] Leibe, B., Seemann, E., and Schiele, B. (2005). Pedestrian detection in crowded scenes. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 878–885.

[Leigh et al., 2015] Leigh, A., Pineau, J., Olmedo, N., and Zhang, H. (2015). Person tracking and following with 2D laser scanners. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 726–733.

[Leistner et al., 2010] Leistner, C., Saffari, A., and Bischof, H. (2010). MI-Forests: Multiple-instance learning with randomized trees. In *Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Proceedings, Part VI*, volume 6316 of *Lecture Notes in Computer Science*, pages 29–42.

[Levinson et al., 2011] Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., and Thrun, S. (2011). Towards fully autonomous driving: systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168.

[Li et al., 2008] Li, Y., Ai, H., Yamashita, T., Lao, S., and Kawade, M. (2008). Tracking in low frame rate video: A cascade particle filter with

discriminative observers of different life spans. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(10):1728–1740.

[Lienhart and Maydt, 2002] Lienhart, R. and Maydt, J. (2002). An extended set of Haar-like features for rapid object detection. In *2002 International Conference on Image Processing (ICIP)*, volume 1, pages 900–903.

[Lin et al., 2007] Lin, H.-T., Lin, C.-J., and Weng, R. C. (2007). A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267–276.

[Linder et al., 2016] Linder, T., Breuers, S., Leibe, B., and Arras, K. O. (2016). On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5512–5519.

[Linder et al., 2015] Linder, T., Girrbach, F., and Arras, K. O. (2015). Towards a robust people tracking framework for service robots in crowded, dynamic environments. In *Assistance and Service Robotics Workshop (ASROB) at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[Liu et al., 2010] Liu, H., Qian, Y., and Lin, S. (2010). Detecting persons using Hough circle transform in surveillance video. In *2010 International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 267–270.

[Liu and Sun, 2009] Liu, H. and Sun, F. (2009). Semi-supervised particle filter for visual tracking. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3928–3933.

[Luber et al., 2010] Luber, M., Stork, J. A., Tipaldi, G. D., and Arras, K. O. (2010). People tracking with human motion predictions from social forces. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 464–469.

[MacCormick and Blake, 1999] MacCormick, J. and Blake, A. (1999). A probabilistic exclusion principle for tracking multiple objects. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999 (ICCV)*, volume 1, pages 572–578.

[Mallick et al., 2002] Mallick, M., Krant, J., and Bar-Shalom, Y. (2002). Multi-sensor multi-target tracking using out-of-sequence measurements. In

Proceedings of the Fifth International Conference on Information Fusion, 2002, volume 1, pages 135–142.

[Maresca and Petrosino, 2014] Maresca, M. E. and Petrosino, A. (2014). Clustering local motion estimates for robust and efficient object tracking. In *Computer Vision – ECCV 2014 Workshops: Proceedings, Part II*, volume 8926 of *Lecture Notes in Computer Science*, pages 244–253.

[Marín-Jiménez et al., 2014] Marín-Jiménez, M. J., Zisserman, A., Eichner, M., and Ferrari, V. (2014). Detecting people looking at each other in videos. *International Journal of Computer Vision*, 106(3):282–296.

[Matthews et al., 2004] Matthews, I., Ishikawa, T., and Baker, S. (2004). The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):810–815.

[Montemerlo et al., 2002] Montemerlo, M., Thrun, S., and Whittaker, W. (2002). Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 695–701.

[Müller et al., 2007] Müller, S., Schaffernicht, E., Scheidig, A., Böhme, H.-J., and Groß, H.-M. (2007). Are you still following me? In *Proceedings of the European Conference on Mobile Robots (ECMR)*, pages 211–216.

[Munz et al., 2010] Munz, M., Dietmayer, K., and Mählisch, M. (2010). Generalized fusion of heterogeneous sensor measurements for multi target tracking. In *2010 13th Conference on Information Fusion (FUSION)*, pages 1–8.

[Orguner and Gustafsson, 2008] Orguner, U. and Gustafsson, F. (2008). Storage efficient particle filters for the out of sequence measurement problem. In *2008 11th International Conference on Information Fusion*, pages 1–8.

[Papageorgiou and Poggio, 2000] Papageorgiou, C. and Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33.

[Pedersoli et al., 2009] Pedersoli, M., Gonzàlez, J., and Villanueva, J. J. (2009). High-speed human detection using a multiresolution cascade of histograms of oriented gradients. In *4th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, volume 5524 of *Lecture Notes in Computer Science*, pages 48–55.

[Pernici and Del Bimbo, 2013] Pernici, F. and Del Bimbo, A. (2013). Object tracking by oversampling local features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2538–2551.

[Platt, 1999] Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74.

[Poschmann et al., 2012a] Poschmann, P., Donner, M., Bahrmann, F., Rudolph, M., Fonfara, J., Hellbach, S., and Böhme, H.-J. (2012a). Wizard of Oz revisited: Researching on a tour guide robot while being faced with the public. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 701–706.

[Poschmann et al., 2012b] Poschmann, P., Hellbach, S., and Böhme, H.-J. (2012b). Multi-modal people tracking for an awareness behavior of an interactive tour-guide robot. In *Intelligent Robotics and Applications: 5th International Conference, ICIRA 2012, Proceedings, Part II*, volume 7507 of *Lecture Notes in Computer Science*, pages 666–675.

[Poschmann et al., 2014] Poschmann, P., Huber, P., Rätsch, M., Kittler, J., and Böhme, H. (2014). Fusion of tracking techniques to enhance adaptive real-time tracking of arbitrary objects. In *The 6th international conference on Intelligent Human Computer Interaction, IHCI 2014*, volume 39 of *Procedia Computer Science*, pages 162–165.

[Reid, 1979] Reid, D. B. (1979). An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854.

[Ross et al., 2008] Ross, D. A., Lim, J., Lin, R.-S., and Yang, M.-H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1):125–141.

[Rowley et al., 1998] Rowley, H. A., Baluja, S., and Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(1):23–38.

[Salti et al., 2012] Salti, S., Cavallaro, A., and Di Stefano, L. (2012). Adaptive appearance modeling for video tracking: Survey and evaluation. *IEEE Transactions on Image Processing*, 21(10):4334–4348.

[Santner et al., 2010] Santner, J., Leistner, C., Saffari, A., Pock, T., and Bischof, H. (2010). Prost: Parallel robust online simple tracking. In

2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pages 723–730. IEEE.

[Schulte et al., 1999] Schulte, J., Rosenberg, C., and Thrun, S. (1999). Spontaneous, short-term interaction with mobile robots. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 658–663. IEEE.

[Schulz et al., 2003a] Schulz, D., Burgard, W., Fox, D., and Cremers, A. B. (2003a). People tracking with mobile robots using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research*, 22(2):99–116.

[Schulz et al., 2003b] Schulz, D., Fox, D., and Hightower, J. (2003b). People tracking with anonymous and ID-sensors using Rao-Blackwellised particle filters. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 921–926.

[Schwartz et al., 1987] Schwartz, J. T., Sharir, M., and Hopcroft, J. (1987). *Planning, Geometry, and Complexity of Robot Motion*, volume 4. Ablex Publishing Corporation.

[Simonnet et al., 2012] Simonnet, D., Velastin, S. A., Turkbeyler, E., and Orwell, J. (2012). Backgroundless detection of pedestrians in cluttered conditions based on monocular images: a review. *IET Computer Vision*, 6(6):540–550.

[Smeulders et al., 2014] Smeulders, A. W. M., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., and Shah, M. (2014). Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(7):1442–1468.

[Smith and Buechler, 1975] Smith, P. and Buechler, G. (1975). A branching algorithm for discriminating and tracking multiple objects. *IEEE Transactions on Automatic Control*, 20(1):101–104.

[Solís Montero et al., 2015] Solís Montero, A., Lang, J., and Laganière, R. (2015). Scalable kernel correlation filter with sparse feature integration. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 24–31.

[Song et al., 2010] Song, X., Zhao, H., Cui, J., Shao, X., Shibasaki, R., and Zha, H. (2010). Fusion of laser and vision for multiple targets tracking via

on-line learning. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 406–411.

[Spinello et al., 2008] Spinello, L., Triebel, R., and Siegwart, R. (2008). Multimodal people detection and tracking in crowded scenes. In *Proceedings of the twenty-third AAAI Conference on Artificial Intelligence*, pages 1409–1414.

[Spinello et al., 2010] Spinello, L., Triebel, R., and Siegwart, R. (2010). Multiclass multimodal detection and tracking in urban environments. *The International Journal of Robotics Research*, 29(12):1498–1515.

[Stalder et al., 2009] Stalder, S., Grabner, H., and Van Gool, L. (2009). Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1409–1416.

[Sudowe and Leibe, 2011] Sudowe, P. and Leibe, B. (2011). Efficient use of geometric constraints for sliding-window object detection in video. In *International Conference on Computer Vision Systems (ICVS)*, volume 6962 of *Lecture Notes in Computer Science*, pages 11–20.

[Supančič III and Ramanan, 2013] Supančič III, J. S. and Ramanan, D. (2013). Self-paced learning for long-term tracking. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2379–2386.

[Tang et al., 2007] Tang, F., Brennan, S., Zhao, Q., and Tao, H. (2007). Co-tracking using semi-supervised support vector machines. In *2007 IEEE 11th International Conference on Computer Vision (ICCV)*, pages 1–8.

[Taylor and Kleeman, 2004] Taylor, G. and Kleeman, L. (2004). A multiple hypothesis walking person tracker with switched dynamic model. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA)*.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT Press.

[Tian and Bar-Shalom, 2009] Tian, X. and Bar-Shalom, Y. (2009). Exact algorithms for four track-to-track fusion configurations: All you wanted to know but were afraid to ask. In *2009 12th International Conference on Information Fusion*, pages 537–544.

[Čehovin et al., 2016] Čehovin, L., Leonardis, A., and Kristan, M. (2016). Visual object tracking performance measures revisited. *IEEE Transactions on Image Processing*, 25(3):1261–1274.

[Viola and Jones, 2004] Viola, P. and Jones, M. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154.

[Viola and Jones, 2001] Viola, P. and Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518.

[Vojíř and Matas, 2014] Vojíř, T. and Matas, J. (2014). The enhanced flock of trackers. In *Registration and Recognition in Images and Videos*, volume 532 of *Studies in Computational Intelligence*, pages 113–136. Springer.

[Vojíř et al., 2014] Vojíř, T., Noskova, J., and Matas, J. (2014). Robust scale-adaptive mean-shift for tracking. *Pattern Recognition Letters*, 49:250–258.

[Volkhardt et al., 2013] Volkhardt, M., Weinrich, C., and Groß, H.-M. (2013). People tracking on a mobile companion robot. In *2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4354–4359.

[Walk et al., 2010] Walk, S., Majer, N., Schindler, K., and Schiele, B. (2010). New features and insights for pedestrian detection. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1030–1037.

[Weinrich et al., 2014] Weinrich, C., Wengefeld, T., Schröter, C., and Groß, H.-M. (2014). People detection and distinction of their walking aids in 2D laser range data based on generic distance-invariant features. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 767–773.

[Wu and Nevatia, 2007] Wu, B. and Nevatia, R. (2007). Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266.

[Wu et al., 2011] Wu, J., Geyer, C., and Rehg, J. M. (2011). Real-time human detection using contour cues. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 860–867.

[Wu et al., 2013] Wu, Y., Lim, J., and Yang, M.-H. (2013). Online object tracking: A benchmark. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2411–2418.

[Xavier et al., 2005] Xavier, J., Pacheco, M., Castro, D., Ruano, A., and Nunes, U. (2005). Fast line, arc/circle and leg detection from laser scan data in a player driver. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3930–3935.

[Xiang et al., 2014] Xiang, Y., Song, C., Mottaghi, R., and Savarese, S. (2014). Monocular multiview object tracking with 3D aspect parts. In *Computer Vision – ECCV 2014: 13th European Conference, Proceedings, Part VI*, volume 8694 of *Lecture Notes in Computer Science*, pages 220–235.

[Yoshikawa et al., 2006] Yoshikawa, Y., Shinozawa, K., Ishiguro, H., Hagita, N., and Miyamoto, T. (2006). Responsive robot gaze to interaction partner. In *Robotics: Science and Systems (RSS)*.

[Yu et al., 2008] Yu, Q., Dinh, T., and Medioni, G. (2008). Online tracking and reacquisition using co-trained generative and discriminative trackers. In *Computer Vision – ECCV 2008: 10th European Conference on Computer Vision, Proceedings, Part II*, volume 5303 of *Lecture Notes in Computer Science*, pages 678–691.

[Yuan et al., 2015] Yuan, J., Chen, H., Sun, F., and Huang, Y. (2015). Multisensor information fusion for people tracking with a mobile robot: A particle filtering approach. *IEEE Transactions on Instrumentation and Measurement*, 64(9):2427–2442.

[Zhang et al., 2005] Zhang, K., Li, X. R., and Zhu, Y. (2005). Optimal update with out-of-sequence measurements. *IEEE Transactions on Signal Processing*, 53(6):1992–2004.

[Zhao and Shibasaki, 2005] Zhao, H. and Shibasaki, R. (2005). A novel system for tracking pedestrians using multiple single-row laser-range scanners. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(2):283–291.

[Zhong et al., 2012] Zhong, W., Lu, H., and Yang, M.-H. (2012). Robust object tracking via sparsity-based collaborative model. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1838–1845.