Automatic Detection of Suspicious Behavior at Railway Crossings

Utrecht University

Master Thesis - ICA-3843475 - Game and Media Technology

Author:
Ronald Dommisse

Supervisor:
dr. ir. Ronald W. Poppe*

External Supervisor:
Dirk Wouters†

Second Examiner:
prof. dr. Albert A. Salah*

August 23, 2020

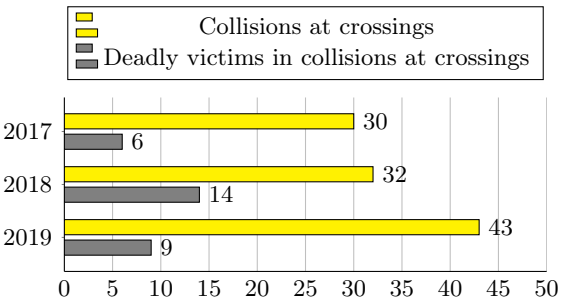
Abstract

In this thesis we study the feasibility of a system to automatically detect suicidal and generally suspicious behavior from surveillance camera feeds in order to assist in the prevention of suicides at railway crossings. We design a system that is able to detect several types of behavior by using computer vision techniques such as object detection and multiple object tracking. We present an approach to extract behaviors from detected person trajectories and to classify these as being suspicious or not, with the intent of sending alerts to camera operators to notify them about railway crossings that need their attention. The aim is to assist these camera operators in their surveillance work. We implement and evaluate the performance of our system and discuss potential areas where improvements to our system can be made.

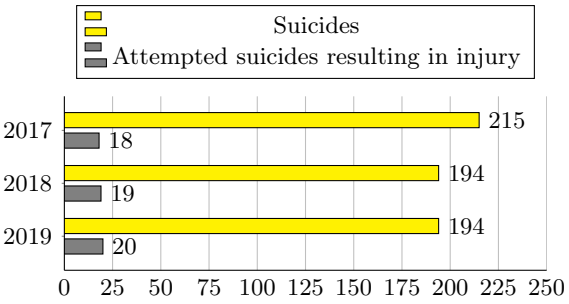
1 Introduction

1.1 Context and motivation

The main railway network in the Netherlands is managed by ProRail, a private company with the State of the Netherlands as its sole shareholder. One of their tasks is to ensure the safety of the railway infrastructure for everyone involved, which includes keeping unauthorized people off the tracks and other dangerous off-limits areas. There are several measures in place for this, one of which is camera surveillance. A major problem that ProRail faces is people attempting suicide by train. There are around 200 such cases each year (see Figure 1). According to ProRail, these attempts occur mostly in somewhat secluded areas along the train tracks. Since the tracks are largely fenced off, the most easily accessible points of entry are railway crossings, which have been found to have a high incidence of suicide attempts [1].



(a) Train-person collisions and deadly victims at railway crossings (excluding suicides)



(b) Suicides and attempted suicides at railway tracks

Figure 1: Train-person collision and suicide statistics in the Netherlands. [2]

There are 2477 crossings along the railway tracks in the Netherlands [2]. A selection of crossings that have been found to have the highest occurrence of suicide attempts are monitored by security staff via overhead cameras. At any time one or two specially trained operators continuously monitor around 15

*Utrecht University
†ProRail

to 20 crossings. In recent years high resolution pan-tilt-zoom cameras have been installed (usually two or more per crossing) that allow the operator to move any camera around as needed. At the time of writing there are 72 cameras in use. This has been a successful strategy for preventing suicides at these particular crossings. There are no official exact numbers, but according to ProRail it is in the order of several tens of prevented suicides per year for this relatively small amount of cameras.

When suspicious behavior is detected by a camera operator, trains on that particular section are immediately informed by control room personnel that an unauthorized person is on the tracks, at which point the train driver will slow down significantly to minimize risk. Authorities are contacted and attempt to arrive on site in time to prevent serious injury.

This manual monitoring is labor intensive and psychologically demanding, which makes it infeasible to keep an eye on every railway crossing at once all the time, so it is not a scalable solution for surveilling all crossings. However, monitoring every railway crossing at the same time would mean the likely prevention of many more suicides, and in turn prevent a serious amount of secondary damage such as psychological trauma in witnesses and disruption of train schedules. Thus, if at all possible, having the capability to monitor all railway crossings could be instrumental to ProRail in reducing the amount of suicides and disruption of train schedules. Since personnel capacity is limited, manual monitoring of such a large amount of cameras is simply infeasible, which means any practical solution would have to be largely automated.

The state of computer vision currently allows for the extraction of a wide array of interesting information from images and videos. We can detect objects, categorize them, track them and even detect spatio-temporal events [3]. We could use these computer vision techniques to develop a system that takes the camera feed as input and returns a simple binary indication whether or not potential suspicious behavior is detected. Many of the sub-problems for this task are already solved, but to our knowledge it has not yet been applied to this specific context. Of course human oversight is still necessary for a final judgment, but if the camera operators only have to look at a camera when an interesting event takes place instead of monitoring it all the time, that would significantly reduce the workload, allowing the operator to oversee more railway crossings at once, and possibly increase the detection rate since there won't be a need to rely upon a human looking at the right camera feed at the right time. Hence, the focus of this thesis will be to research the feasibility of a system to aid camera operators in detecting suicidal behavior at railway crossings as early as possible. More specifically, the novel aspect will be analyzing and classifying behavior by making use of existing techniques.

1.2 Suspicious behavior

While there are many types of behavior at railway crossings that would be considered suspicious, such as putting coins on the tracks, taking frontal photos of trains, etc., the focus of this thesis will primarily be on detecting early signs of suicidal behavior. However, since our method is aimed at detecting behavior that is simply out of the ordinary, many of these other types of suspicious-but-not-suicidal behavior will also be picked up.

There has been some research into suicidal behavior at train tracks [1, 4–6], but it appears that many of the cues are vague and often the gut feeling of camera operators plays a significant role in the process of deciding what is suspicious, as revealed by interviews with ProRail personnel. Many of these vague cues could conceivably be detected by an AI, but this is outside the scope of this thesis. One of the more objectively measurable cues is the trajectory of a person. Combined with information about the scene, for example the barriers being open or not and the location of train tracks, this gives a significant amount of information, such as a person waiting while no train is coming, a person walking onto the tracks, etc. With this limited information we will not be able to perfectly detect suspicious behavior, but even a rough indication of anything out of the ordinary is useful according to ProRail. Hence, we will primarily focus on using the trajectory of persons and environmental cues to determine whether their behavior is suspicious.

1.3 Research questions

The main goal for ProRail is to find out whether an automated monitoring system is feasible. Hence, in this thesis our main research question is:

> *Is a real-time automated system to detect suspicious behavior at railway crossings technically feasible?*

To make this more concrete, we break it up into more specific sub-questions:

1. *What is the accuracy of detection and tracking of people?*
2. *What types of behaviors can we detect based on trajectories and how well can we detect these?*
3. *How quickly can we detect these behaviors?*

We determine the feasibility of a real-time automated vision-based system for analyzing and classifying suspicious behavior at railway crossings. To this end we perform a literature study (Section 2) and design a general method (Section 3) for which we develop a proof of concept. We evaluate the performance of person detection, person tracking and behavior analysis in our proof of concept by defining metrics and discussing the results (Sections 4 and 5). Finally, we will discuss potential future work (Section 6).

2 Literature review

We review literature regarding three key aspects in our task of suspicious behavior detection: object detection, object tracking and detection of anomalous behavior.

2.1 Object detection

Object detection is the task of detecting and localizing classes of objects in images or sequences of images to gain semantic information about the image. It has found widespread use in a variety of domains as one of the fundamental building blocks of computer vision. Some examples of the wide array of application domains are surveillance, face recognition, optical character recognition, pedestrian detection and vision systems for autonomous vehicles.

History One of the earliest real-time object detectors was the Viola-Jones algorithm [7], which relies on hand-crafted Haar features to recognize semantically meaningful structures in images. Other early approaches are Scale Invariant Feature Transform (SIFT) [8] and Histogram of Oriented Gradients (HOG) [9], which give feature descriptors that can be used with machine learning techniques, for example a Support Vector Machine (SVM) [10], to train a classifier that can detect objects.

These approaches follow a traditional division of sub-tasks in object detection:

- Region selection, where areas of an image that may contain an object are selected. This is usually done with a sliding window approach that scans the whole image.
- Feature extraction, where some abstract representation of the object in the region of interest is computed.
- Classification, where a machine learning method is trained to discriminate between feature representations in order to classify the detected object as belonging to a certain class of objects.

In recent years, object detectors based on Deep Neural Networks (DNNs) [11] have exploded in popularity due to the widespread resurgence of neural networks, in particular the Convolutional Neural Network (CNN) [12]. Although the concept of artificial neural networks has been around since at least the 1940s, research stagnated after the 1960s as it was deemed an infeasible approach due to the computational cost and large amounts of data necessary to train neural networks. With the introduction of back-propagation [13] in the 1980s, popularity blossomed before declining again in the early 2000s. By the late 2000s, breakthrough research [14, 15] had renewed the interest in deep learning, as the availability of large data sets such as ImageNet [16] and the increased availability of massively parallel computing power meant that it was now feasible for DNNs to be competitive with traditional object detection methods, as they could be trained on regular hardware.

Neural networks Where the above mentioned traditional models create a shallow hierarchy from pixels to abstract features, limiting the amount of semantic information that can be detected, CNN based methods create much deeper hierarchies of features, allowing for vastly more semantic expressiveness. CNNs learn feature representations directly from the source data, eliminating the need for manually crafted features, which were a limiting factor in the traditional models. Because of this, they have a much greater capability for reducing the dimensionality of computing problems [17].

Generally, there are two main types of architecture within CNNs. First, there are those that mirror the traditional approach where first regions of interest are generated, which are then classified into different categories. Examples of these types of networks are R-CNN, Fast R-CNN, Faster R-CNN, SPPnet, FPN and Mask R-CNN. The other type of architecture does away with the two-stage process and instead opts to unify the localization and classification of object into a regression problem. These include for example AttentionNet [18], SSD and YOLO.

R-CNN [19] uses selective search [20] to generate 2000 region proposals, then uses a CNN to extract deep feature representations for each region proposal, after which linear SVMs are used to score these feature representations. These scores are used in a bounding box regression problem to adjust the regions, after which non-maximum suppression is applied to filter the resulting bounding boxes.

While R-CNN's detection performance is a significant improvement over traditional methods, it comes at a high computational and space cost. It is slow to train and the selective search for region proposals takes too much time for the method to

work in real-time.

SPPnet [21] improved on the performance of R-CNN by applying Spatial Pyramid Matching [22]. Increased speed over R-CNN was achieved by extracting only a single feature map instead of the expensive region proposal step in R-CNN.

Fast R-CNN [23] improves on R-CNN and SPPnet with a more efficient training method that takes advantage of feature sharing during training. Like SPPnet, instead of generating 2000 region proposals and doing a forward pass for each of them, images are fed directly to the CNN to create a convolutional feature map. The region proposals and an image are now processed in a single forward pass. Fast R-CNN introduces a region of interest (RoI) pooling layer (similar to a single layer SPPnet) into which the proposals are fed, which is then sent into a fully convolutional layer for both classification and bounding box regression. The SVM for classification is replaced with a softmax layer.

The bottleneck in Fast R-CNN is the selective search for region proposals. Faster R-CNN [24] addresses this by replacing the selective search with a Region Proposal Network (RPN). A convolution feature map is generated and fed into the RPN to generate anchors which are fed into the classification and bounding box regression layers.

R-FCN [25] introduces position-sensitive score maps and moves the fully convolutional layers to before the RoI pooling, avoiding the need to apply a per-region subnetwork hundreds of times as in Fast/Faster R-CNN

In [26] the Feature Pyramid Network (FPN) is proposed as a generic feature extractor with an architecture consisting of a bottom-up pathway and a top-down pathway with lateral connections, combining low resolution semantically strong features with high resolution semantically weak features. FPN can replace the feature extraction in other architectures such as Faster R-CNN.

Mask R-CNN [27] extends Faster R-CNN by concurrently predicting object masks and bounding boxes, allowing pixel-level instance segmentation. Additionally, Mask R-CNN can be generalized to tasks such as human pose estimation.

Handling all the different components such as region proposal generation, feature extraction, classification and bounding box regression separately is a bottleneck in real-time applications [17]. Single-shot detectors unify these steps to reduce computational cost.

YOLO (You Only Look Once) [28] unifies bounding box prediction and classification in a single neural network, directly predicting class probabilities. An image is divided into a grid where object bounding boxes and confidence scores are predicted for each grid cell. Detecting objects now requires only a single pass through the pipeline, allowing for extremely fast processing speed, at the cost of accuracy, as YOLO has difficulty dealing with small objects.

Single Shot MultiBox Detector (SSD) [29] extracts features using a VGG16 network [30] and uses convolutional layers to make independent object predictions from convolutional maps at different scales. Similarly to YOLO, SSD discretizes the image space, but unlike YOLO it does this at different aspect ratios and scales, leading to better handling of objects of various sizes.

YOLOv2 [31] improved upon YOLO by introducing batch normalization [32], anchor boxes, multi-scale training, increased input resolution and using smaller grid cells to better handle smaller objects for increased accuracy and speed, as well as being able to take different input image sizes, where lower input sizes lead to faster processing but lower accuracy.

YOLOv3 [33] made incremental improvements to YOLOv2, including the use of a deeper neural network architecture for increased accuracy at the cost of more processing speed. Predictions are done at 3 different scales, making it better at detecting small objects. Objectness confidence is predicted using logistic regression. Classes are predicted using logistic classifiers, replacing the softmax classification of YOLOv2.

Pedestrian detection Much research has been done regarding the specific application of object detection for pedestrian detection [34], for example with approaches based on Haar-wavelet cascades with AdaBoost [35], HOG with linear SVM [9] and Neural Networks using Local Receptive Fields [36]. In recent years methods combining deep learning with classical machine learning models seem to yield the most successful results [37].

In [38] the use of Faster R-CNN for pedestrian detection is investigated and an approach to pedestrian detection is proposed based on a combination of RPN for region proposals and features and a boosted forest [39] for classification.

Faster R-CNN based approaches are too slow for real-time applications though, so [40] proposes a single-stage pedestrian detection architecture called ALFNet based on SSD by introducing Asymptotic Localization Fitting, in which the anchor boxes of SSD are evolved by a series of predictors to improve detection results.

Occlusion awareness Occluded objects can be difficult to detect. [41] proposes an add-on to Faster R-CNN to handle partially occluded pedestrians by employing an attention mechanism across the CNN channel features to represent various occlusion patterns in a single model, where occlusion patterns are treated as specific combinations of body parts. This allows for prediction of full bounding boxes for only partially visible pedestrians.

Object trackers may learn incorrect object appearance models when the object is occluded. [42] proposes an occlusion detection method based on an object's color profile to prevent a tracker from updating when the object is occluded.

An extension to single-stage detectors (e.g. YOLO, SSD) for better handling of occluded pedestrians is proposed in [43], where the the output layers of the neural network are updated to include the prediction of part confidence scores, leading to an occlusion-aware detection score and reducing false negative detections. Secondly, a post-refinement classifier is introduced to reduce false positive detections.

In [44] a method is proposed for simultaneous pedestrian detection and occlusion estimation by employing a CNN consisting of two branches that regress bounding boxes for full persons and visible parts of persons, respectively. These two branches complement each other to improve detection performance.

Pose estimation In addition to just detecting and localizing persons, it can be useful to also know their pose. Techniques such as DeepCut [45], DeeperCut [46] and Mask R-CNN allow for detection of keypoints in objects, e.g. body parts, that can be used to construct structural representations of objects, allowing for the estimation of the pose of multiple humans in a scene. DensePose [47] goes even further by estimating dense human poses that represent the surface of a person.

2.2 Object tracking

Object tracking refers to the process of inter-frame association of objects, e.g. following objects throughout a video sequence. For our purposes we will only consider the case of multiple object tracking (MOT). We consider tracking in screen space, while for accurate passive object tracking in 3D depth cameras or stereo cameras are necessary [37].

Tracking by detection Tracking by detection seems to be the most popular approach in multiple object tracking, where the tracking system is separate from the detection system [48].

Simple Online and Realtime Tracking (SORT) [49] is one of such approaches where the object detection part is delegated to an external process and instead opts to focus solely on inter-frame association of detections. It is aimed at real-time applications, as opposed to batch-processing based object tracking methods such as [50–59], utilizing a CNN as a detection framework. The key components of this approach are a Kalman filter [60] to propagate track bounding boxes to the next frame, approximating their motion with a constant velocity model, and a data association step based on the Hungarian algorithm [61] to optimally match detections to existing tracks by building an assignment cost matrix from a (gated) intersection-over-union (IoU) distance between detections and predicted bounding boxes. With this approach short term occlusions can be handled, as the Kalman filter will make continued predictions while a track has no associations and the IoU distance helps to match only detections with similar scale. New track hypotheses are initiated for any unmatched detections. These are initially considered tentative until several subsequent detections have been associated with the track to mitigate the influence of false positive detections. Tracks are deleted if no detections are associated with it for a certain amount of frames.

Deep SORT [62] extends the SORT algorithm with a deep association metric to incorporate appearance information in the matching process to reduce identity switches, in addition to the motion information handled by the Kalman filter. Motion information is incorporated into the assignment cost matrix by the use of a (thresholded) Mahalanobis distance [63] to compare predicted Kalman states with detections. Additionally, appearance information is incorporated into the cost matrix by extracting an appearance descriptor from detection patches. A limited history of these descriptors is kept for each track. When computing the cost of matching a detection with a track, the (thresholded) minimum cosine distance between the detection's appearance descriptor and the track's descriptor history is taken. This appearance based matching complements the motion based matching and aids in recovering from long-term occlusions. The motion and appearance metric are then combined into an assignment cost using a weighted sum.

The appearance descriptors are extracted by a CNN trained to extract features that can be used for nearest neighbor queries using the cosine similarity metric [64].

The matching process is done through a cascade of smaller problems, where detections are assigned to tracks by solving the assignment problem using the Hungarian algorithm in order of increasing track age. Detections that remain unmatched after this are assigned by IoU matching as in the original SORT algorithm.

Recurrent YOLO [65] combines YOLO with Long Short Term Memory (LSTM) [66] to extend YOLO into the spatiotemporal domain. Objects are detected using YOLO and, together with preliminary location inferences, these are then sent to an LSTM module at the end of the pipeline for tracking.

Tracktor++ [67] takes a different approach and uses the bounding box regression part of an object detector (specifically Faster R-CNN) to predict

the object positions in the next frame, turning a detector into a *tracktor*. This only works when objects change only slightly from frame to frame. To compensate for camera motion and low frame rates this tracktor is extended with a motion model consisting of camera motion compensation and a constant velocity assumption. Additionally, short term re-identification is applied using appearance features.

Moving away from the tracking by detection paradigm and instead opting to integrate detection and appearance embedding in a shared model, the Joint Detector and Embedding model (JDE) [68] introduces a near real-time single-shot deep architecture to output detections and appearance embeddings simultaneously using a single neural network based on FPN.

2.3 Behavior detection

In the context of our problem domain, we are interested in finding suspicious behaviors in surveillance videos. We discuss two types of techniques that are relevant to this task: action recognition and anomaly detection.

Action recognition Action recognition refers to the process of detecting and classifying actions in image sequences. [69] and [70] define action primitives as atomic movements that can be described at the limb level. Then actions are defined as a collection of action primitives that describe a, possibly cyclic, whole body movement. Activities are defined as a number of subsequent actions.

After a period where holistic representations of motion such as [71–73] were popular, in recent years local and deep representations are favored instead [74].

In local representations spatiotemporal interest points are first detected [75–78], from which local descriptors are extracted, for example edge and motion descriptors [79–82] or pixel pattern descriptors [83–88], which are then aggregated to build discriminative descriptors [82, 89–94].

[74] places deep learning architectures for action recognition into four different categories: spatiotemporal networks, multiple stream networks, deep generative networks and temporal coherency networks.

Spatiotemporal networks extend CNN architectures into the temporal domain by using 3D kernels that operate on several adjacent frames [95–100] to obtain descriptors that encode spatiotemporal features.

Multiple stream networks are CNNs that, inspired by the human visual cortex that processes object attributes and object motion through separate pathways [101], handle appearance and motion information in separate streams [102, 103].

Deep generative networks [66, 104–108] seek to learn temporal structures in data without supervision, which is useful when there is little to no labeled training data.

Temporal coherency networks [109–113] exploit the assumption that motion and appearance do not change abruptly between consecutive frames.

Anomaly detection In the context of video surveillance, anomaly detection is the process of identifying and localizing events that do not conform to the norm. It can be considered an unsupervised pattern recognition task [114]. Practical examples of such tasks include detecting fights in crowds [115] or traffic accidents [116].

Due to the sequential nature of video, deep learning architectures suitable for anomaly detection in video surveillance include CNNs, Recurrent Neural Networks (RNNs) and LSTMs [117]. Other deep learning based methods include generative models such as Variational AutoEncoders [118] Generative Adversarial Networks [105] and Adversarial AutoEncoders [119] [114].

Since anomalies are by definition rare, it is usually difficult to obtain positive examples. Therefore, state-of-the art techniques often learn reference models of the normal situation in an unsupervised way and compare regions of interest containing potentially anomalous events with these during testing, considering anything that deviates an anomaly. Properties of these regions of interest can be represented in multiple ways. HOG and Histogram of Optical Flow (HOF) [81] have been used to model spatiotemporal properties of trajectories in video data, but these cannot handle occlusions and have high complexity [120]. CNNs have been successfully employed in image classification, object detection and action recognition, but they are slow in processing speed and since it is a form of supervised learning, large amounts of (positive and negative) training data are necessary, which is a challenge when positive examples are rare.

There has been much research into optimizing CNNs so they can be of more practical use. [120] proposes a real-time method for anomaly detection and localization in crowded scenes using fully convolutional neural networks (FCNs). Using transfer learning, a pre-trained supervised CNN is transferred into an unsupervised FCN for generating and describing abnormal regions in videos.

It may be difficult or impossible to to define a normal event that takes into account all possible normal behaviors [121], making it incorrect to assume that all detected behaviors that deviate from what is learned to be normal are an anomaly. Moreover, the boundary between what behavior is anomalous and what is normal is often ambiguous, which poses a challenge for both conventional machine learning

and deep learning approaches [122].

In [123] a deep learning approach to detect real-world anomalies in surveillance videos is proposed. Annotation is done at video level, eliminating the need for temporal annotations, which is a costly process. Videos are treated as bags of fixed-length segments where at least one segment in a positive bag $\mathcal{B}_a$ is assumed to contain an anomaly. In a negative bag $\mathcal{B}_n$ none of the segments contain an anomaly. A deep multiple instance learning (MIL) model is applied to achieve a ranking between an anomalous video $\mathcal{V}_a$ and a normal video $\mathcal{V}_n$ using a multiple instance ranking objective function:

$$\max_{i \in \mathcal{B}_a} f(\mathcal{V}_a^i) > \max_{i \in \mathcal{B}_n} f(\mathcal{V}_n^i)$$

where $f(\mathcal{V}_a^i)$ and $f(\mathcal{V}_n^i)$ are the predicted anomaly scores for each video segment and the maximum score is taken over all segments in each bag, enforcing ranking only on the two segments with the highest score from each bag. Together with a sparseness constraint to account for the fact that anomalous segments in a bag are sparse and a smoothness constraint to make the anomaly score vary smoothly between segments, this forms a MIL ranking loss function $l(\mathcal{B}_a, \mathcal{B}_n)$, which is used to create a complete objective function $\mathcal{L}(\mathcal{W}) = l(\mathcal{B}_a, \mathcal{B}_n) + \lambda_3 ||\mathcal{W}||_F$, where $\mathcal{W}$ represents model weights. This objective function can be used by a deep learning network to learn anomaly scores for video segments.

3 Methodology

In this section we discuss the data that is available to us for development and testing and present a design for a system to detect suspicious behavior. The approach can roughly be divided into 3 main steps:

- Detection of objects (persons, barrier states, etc.).
- Tracking persons based on detections.
- Analysis of tracks to extract behaviors and classifying these as suspicious or not.

3.1 Available data

ProRail has high resolution pan-tilt-zoom (PTZ) cameras installed on their railway crossings, usually from at least two different perspectives. This means a high resolution video stream is available for analysis. Since the camera parameters are static most of the time (unless an operator is manually moving it and thus also looking at the camera feed already), we have not concerned ourselves with dealing with moving cameras and have assumed a static camera orientation and zoom level for the purpose of this thesis. To limit the scope of this thesis we designed our system to work with a single camera feed and mostly in screen space, rather than world space. For the development of our system, the available data consists of 1080p 25 fps video files of a single railway crossing. See Section 4.3 for data set extracted from these videos that we use for evaluation.

3.2 List of suspicious behaviors

We identified a set of potentially suspicious behaviors which could be detected based on the trajectory of persons. These are:

- Waiting while the barriers are open.
- Walking back and forth while the barriers are open (i.e. turning and moving significantly into another direction).
- Walking onto train tracks outside the permitted area (see also Figure 4).
- Being on the train tracks while the barriers are closed.

In order to detect these relatively abstract behaviors, we first focus on detecting a set of more rudimentary behaviors which we can use to reason about the intentions of a person:

- Start/stop moving.
- Enter/leave train tracks.
- Enter/leave forbidden area.
- Change direction (as well as moving a significant distance into another direction).

3.3 Approach

To determine whether an automated system for detecting suspicious behavior is possible, we design and implement a proof of concept, of which we first give a general outline.

3.3.1 Requirements and general design

The goal is to track people in a video stream and analyze their trajectories in real time. Currently, the prevailing paradigm for tracking objects is tracking-by-detection ([62]), for which we need an object detector in addition to an object tracker. We also need some information about the environment, such as the state of the barriers, for which we can use an object detector as well. We design the system in a modular way so that different components can be substituted according to different needs (e.g. trade-offs between speed and tracking performance). After we have acquired the trajectories of persons, we need

to analyze and evaluate their behaviors, for which we design a novel module.

We divide the system into 4 main modules (see also Figure 2):

- An environmental state detector.
- A person detector.
- A person tracker.
- A behavior analysis and classification module.

We discuss these components in more detail in the subsequent sections.

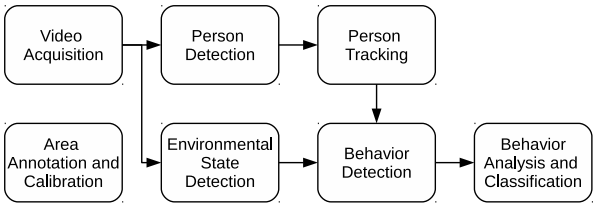


Figure 2: High level design outline

3.4 Environmental state detector

The purpose of the environmental state detector module (ESD) is to infer information about the scene that is relevant to classifying the behavior of persons and present it in a way that is usable in the behavior detection and classification module. For our purpose we limit this functionality to detecting the state of the barriers. We consider 4 states: closed, opening, open and closing.

Like the person detector (see the next section), this module is built on an object detector (specifically Tiny YOLOv3 [33]). While we discriminate between opening and closing, the object detector cannot see the difference, so the network is trained to detect 3 different states: open, closed and transitioning. Annotating the training data is done with a specially built tool (see Section 4.2).

Each frame, the object detector gives a list of detected barriers in a certain state. Since all barriers in the scene should have the same state, we sum all detections per state weighted by their confidence score and use the highest scoring state as the actual state. We keep track of the previous state to determine whether a transitioning barrier is either opening or closing. We smooth this transition process by requiring successive detections of a certain state for a set amount of time before the transition is finalized. This is done by gradually increasing a score every frame when a different state than the previous accepted state is detected *and* the currently detected state is equal to last frame's detected state. If these conditions are not true the score is decreased. When the score reaches a threshold, the change is accepted and the score reset to zero. This increment and threshold mechanism is used in other components of our system as well, as will be shown later. This approach makes the barrier state detector robust to random detection errors.

3.5 Person detector

The purpose of this module is to detect people in the scene, either on foot or on a bicycle, and return a bounding box for each person. We selected YOLOv3 [33] with a network pre-trained on the MS-COCO data set [124] as our object detector because of its speed and relatively good detection rates, making it highly suitable for real-time applications. While the MS-COCO set contains a large variety of object classes, only the relevant classes are considered and the rest is filtered out after a forward pass of the network. Non maximum suppression is applied to filter out duplicate detections. The final output of this module is a list of detections consisting of a bounding box, a label and a confidence score for each detection.

Ignored detections Clearly visible persons are detected well enough, but highly or often obscured persons (such as those on the other side of the road, intermittently occluded by vehicles) are not detected very consistently by YOLOv3. Since there are multiple cameras per crossing, we assume these persons will be better visible on one of the other cameras, so we filter out all detections that are in low-visibility areas by checking whether they are within an area that is manually annotated to be ignored (see Figure 4). This reduces erratic detections which helps the tracker and allows us to focus more on quality tracks for behavior analysis.

Bicycles and dogs It can be useful to know if detected persons are cyclists or are walking a dog, since these could possibly provide clues to the intentions of people. To this end we use the person detector to also detect dogs and bicycles. Detected bicycles and dogs are matched to the nearest person. Since these objects are often rather small and their detection is not very robust, there are many false negatives. To counteract this we keep track of confidence scores for each person indicating whether this person is a cyclist or has a dog, respectively. These (clamped) scores decrease each frame as long as there are no detections, otherwise they are increased proportional to the confidence of the detection.

3.6 Person tracker

To associate detections between frames to the same person we employ an object tracker. We selected Simple Online and Realtime Tracking with a Deep

Association Metric (Deep SORT) as the tracker in our system [62] since it is fast, simple, and has an implementation readily available. It tracks objects in screen space based on detected bounding boxes and only needs a single camera feed, so it is suitable for our needs. Since it was explicitly developed with person tracking as a use case, in addition to the tracker implementation, a pre-trained network for feature extraction from image patches containing persons is also available, which is exactly what we need.

Deep SORT deletes tracks as soon as they go inactive (when it goes for more than a certain amount of subsequent frames without any matched detections), so our tracker consists of a layer on top of it that manages the tracks that Deep SORT outputs. The output of Deep SORT is simply a bounding box for every track at each frame. We are interested in the sequence of all bounding boxes of a track, so we record the output of Deep SORT in our track manager. A history of all tracks is maintained along with additional information for each track, such as whether a dog or bicycle was associated with the track.

For each frame of a track, along with the bounding box, we record whether the sample was the result of a detection, a prediction (by the Kalman filter in Deep SORT) or an interpolation, as we delete the previously predicted samples and replace them with interpolated samples whenever we encounter a new detected sample. While the predictions are useful when there is no future information about track's position, filling gaps between detections with interpolations yields more accurate results. To this end we modified Deep SORT to indicate whether a track sample was the result of a matched detection or a prediction.

We also record whether the sample was occluded or not (ideally this should be determined by some kind of occlusion detection mechanism. However, we currently determine this by whether the sample was the result of a detection or not).

This approach makes it possible to later extend the person tracker with more functionality, such as smoothing of the trajectories or matching broken tracks together, should the need arise.

After an update step, the tracker contains a list of tracks with an indication for each track whether it is currently active and when it was last updated. Due to the way we implemented the update mechanism, each track can contain a segment of predicted samples at the "head" of the track, although these predictions are replaced when a new detection is matched to the track. Hence, when designing our real-time behavior analysis system, we must consider the fact that the newest samples might not be accurate, and that samples in the recent past might change.

3.7 Behavior Analysis and Classification

Given a set of person trajectories in screen space, we present a method to analyze their behaviors. We roughly divide the approach into three parts. First, we determine basic behaviors that are more or less directly measurable: the velocity of a person, whether a person is moving or not and if they are currently located in any special zone. From these basic behaviors we determine more complex behaviors that become apparent over time, such as a person changing direction or a person waiting for a prolonged period of time for open barriers. Finally, we analyze the detected behaviors and classify them to achieve a binary indication of whether a person is behaving suspiciously or not. Changes in behavior are handled by an event manager that raises an alert if something suspicious happens. See Figure 3 for a schematic overview.

3.7.1 Basic behavior

Velocity estimation In order to determine whether a person is moving we need to first determine the velocity $\mathbf{v}_t$ at time t. While the trajectories are measured in screen space, directly using distances in pixels to compute the velocity does not work very well, as it gives skewed results due to camera perspective and distortion. Instead, we use a calibrated model of the camera (see Section 3.8) to transform the screen space coordinates to world space to compensate for perspective.

As the projection from screen space to world space is not a one-to-one mapping, the result of this is a ray instead of a point. We determine the (3D) world space position by intersecting this ray with a virtual plane. Since we use the centroids of bounding boxes for positions and assume that people are roughly 1.8m high, we place this virtual plane parallel to and at 0.9m above the ground. We assume all persons are on the ground, so we ignore the height component of the (world) position and proceed in the 2D plane. Together with the known frame rate the world space position is used to convert the displacement from pixels per frame to an approximation in meters per second.

The velocity $\bar{\mathbf{v}}_t$ at frame t is then defined as a moving average over the last n_f frames:

$$\bar{\mathbf{v}}_t = \frac{1}{n} \sum_{i=t_0}^{t} \mathbf{v}_i \,, \quad t_0 = \max(0, t - n_f) \,, \quad n = t - t_0$$

This approach functions as a form of low pass filtering and is aimed at getting a better representation of the general direction the person is heading in than by just computing the velocity from the displacement between the last two frames.

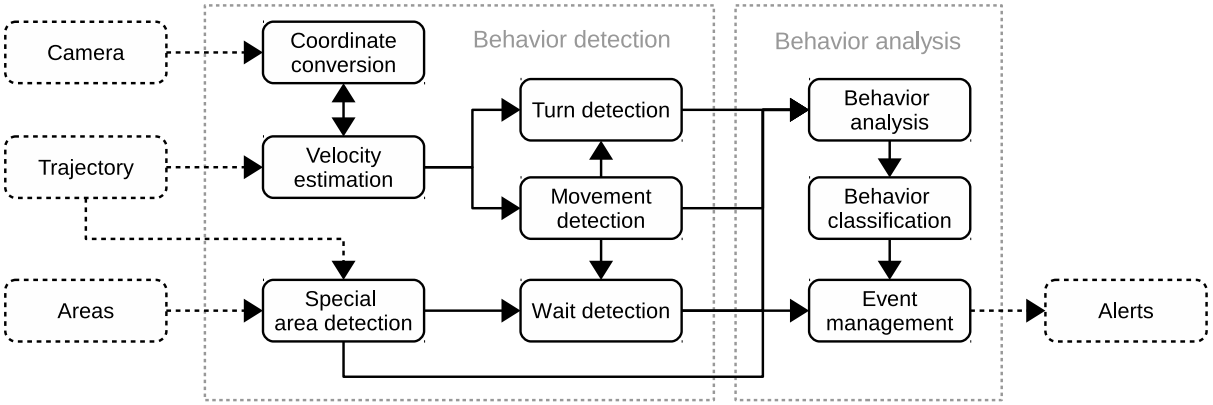


Figure 3: Schematic overview of the behavior detection and analysis module.

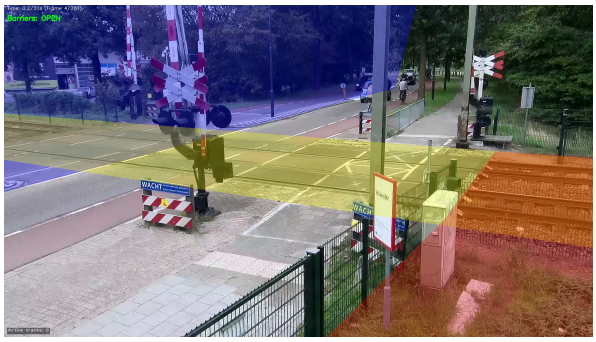


Figure 4: Manually annotated areas. Blue: ignored detections, yellow: train tracks, red: forbidden area.

Movement A binary indication whether a person is moving is determined as follows. Every frame the state of the person's behavior is updated. If the speed at time t exceeds a certain threshold s_{min} we increase a variable $move\_score$ by some predetermined rate r_{move}. If it does not we decrease it by r_{stop}. In both cases $move\_score$ is clamped to $(0, 1)$. If $move\_score$ hits 0 or 1 and the flag is_moving is true or false respectively, then is_moving is flipped. A history is kept of the state of is_moving at each frame.

Presence in special areas To get a binary indication of a person being on the train tracks or in a forbidden zone (e.g. on parts of the track that are not part of the crossing), we first manually annotate these special areas as polygons on the screen (see Figure 4). We then check whether the bottom center of the bounding box of the trajectory is inside one of these polygons (under the assumption that this point most accurately represents the position of the feet of the person in question) and if so, we increase the score for the person being in that type of area by a rate r_{enter}. If the person is not in a specific area type, the score for that area type is decreased by r_{leave}. In either case the score is clamped to $(0, 1)$. If the score hits 0 and the flag for being in that area is true, we know the person has left the area and flip the flag to false. Likewise, if the score hits 1 and the flag is false, we know the person entered the area and flip the flag to true.

3.7.2 Complex behavior

Turn detection The method used to detect whether a target has (significantly) changed direction is outlined in Algorithm 1. For every track, we keep a running average $\bar{\mathbf{d}}_c$ (lines 5-7) and a moving average $\bar{\mathbf{d}}_m$ (lines 8-15) of the direction over the last n_{lb} frames. We only update these when the target is moving (obtained previously, as in Section 3.7.1) and for the moving average we only look back as long as the target is was moving. We do this to filter out the noisy velocity measurements when a target is not moving (or not moving much). With this approach, $\bar{\mathbf{d}}_c$ represents the average direction before the target stopped moving and $\bar{\mathbf{d}}_m$ the average direction after the target started moving again.

We then take the angle θ between $\bar{\mathbf{d}}_c$ and $\bar{\mathbf{d}}_m$ and compare it to a threshold angle θ_{max} (lines 16-20). If the threshold angle is exceeded, then we know a significant change in direction has occurred.

To smooth out jitter we then gradually increase a "turning score" s, so that after δ seconds of successive turn detections we consider the target to have turned, at which point we reset s and $\bar{\mathbf{d}}_c$ so that the running average corresponds to the current direction again (lines 21-32).

An illustration of this process can be seen in Figure 5

While direction changes while a person is standing still are ignored, this approach is not taking into account how far a person moves. To filter out detected turns where a person only moves around slightly, we can set a minimum distance d_{min} for a person to move before we consider the change in direction significant.

The process for this is as follows: keep a binary flag $turn\_queued$ that is initialized to false. When a

Figure 5: Turn detection

Illustration of the turn detection algorithm in action. The top graph shows the angle of the velocity over time with the frames where the target was not moving shaded in red. The bottom graph shows the running average and the moving average of the direction, as well as the "turning score" and a thick vertical line when the score reaches 1 and a turn event is issued. The areas shaded in light red are frames where the angle between the running and moving average exceeded the threshold. Note that these values are circular (angles wrap around after 2π), hence the values at the vertical extremes of the graphs are the same.

turn is detected by Algorithm 1 and *turn_queued* is false, remember the current world space position p_0 and the average direction $\bar{\mathbf{d}}_0$ *before* the turn and set *turn_queued* to true. Subsequently, if *turn_queued* is true, the distance from the current position to $p_0 > d_{min}$ and the angle between the current direction and $\bar{\mathbf{d}}_0 > \theta_{max}$, we issue the turn detected event and reset *turn_queued* to false.

We can then combine the detection of a person changing direction with the state of the barriers to find out if a person is moving back and forth while the barriers are open, which is suspicious.

3.7.3 Classification

From all the basic and more complex behaviors we detected, we need to determine whether a person is behaving suspiciously enough to warrant attention.

Initially, it might make sense to bundle all the behaviors into some kind of "suspiciousness score", allowing for an ordering in the suspiciousness of different combinations of behaviors to allow for some sort prioritization where the "most suspicious" events will be given precedence in a system with many camera feeds, always directing the camera operators towards the scenes that need the most attention at any given time.

However, it it difficult to obtain such a score for every combination (and duration) of events, so we limit ourselves to a strictly binary classification to indicate whether a scene warrants attention or not.

Every frame each person is checked for certain combinations of state and actions. The following combinations result in a high alert event being triggered:

- A person has been inside a forbidden zone for more than a certain amount of time.
- A person has not moved for a certain amount of time while the barriers are open.
- A person has significantly changed direction more than a certain amount of times (regardless of barrier state).
- A person significantly changed direction while the barriers were open.
- A person is on the train tracks while the barriers are closed.

The high alert event is triggered only once, after which a cool down period is entered where no new high alert events can be triggered for that person. This is to avoid sending out a flood of alert messages, as it should be enough to send an alert only once (with a timestamp indicating when the suspicious behavior was detected).

3.7.4 Event management

The purpose of this module is to collect the various behavior events, log them and send out alerts to external systems that need to be notified of suspicious behavior.

3.8 Camera calibration

In order to transform image space coordinates to world space coordinates, we need a virtual model of the real camera, containing its intrinsic and extrinsic parameters. The intrinsic parameters encompass sensor dimensions and focal length, as well as lens distortion. Extrinsic parameters describe the position and orientation of the camera in world space.

We use [125] to model our camera. When parameters are known these can be set directly, as is often the case with intrinsic parameters. In the case of unknown extrinsic parameters, a collection of points of known height can be used to fit a model of the parameters using Metropolis Monte-Carlo sampling.

For example, we use the height of person detections and assume they are 1.8m tall with a standard deviation of 0.1. Additionally, we manually indicate a horizon line to help the fitting. Since we assume a static camera position and orientation, this calibration step only needs to be done once. See also Figure 6.

Figure 6: Extrinsic camera parameters established by providing a rough estimation and subsequently using the known height of person detections to fit a more accurate model. Visualized by a grid of orthonormal direction arrows on the ground plane in world space.

Generally, both intrinsic and extrinsic parameters are known for surveillance cameras, thus the model can be correctly calibrated a priori without the need for the additional parameter fitting step. In the case of PTZ cameras, although their extrinsic and intrinsic parameters are variable (e.g. a varying orientation and focal length, respectively), these tend to be known by some feedback system, thus they can be integrated into our method to update the camera model on the fly.

4 Experiments

In this section we describe our software implementation of the system and discuss the parameters that were used (Section 4.1), describe the data annotation process (Section 4.2), present the evaluation data set (Section 4.3), define metrics for quantitative evaluation (Section 4.4) and describe our evaluation methodology (Section 4.5).

4.1 Implementation

We briefly discuss our software implementation and list the parameters used for the various components.

4.1.1 Experimental software platform

To experiment with different techniques and develop a proof of concept we developed a software implementation of our system in Python. Many computer vision and machine learning techniques have readily available implementations in the form of Python libraries, which enabled relatively quick development of features in our system. We make use of OpenCV [126], particularly the DNN module, which has a Darknet [127] backend that we use to load the YOLOv3 models for our object detectors. Video acquisition and processing is also provided by OpenCV. Additionally, we use the implementation of Deep SORT provided by the authors on GitHub, which uses TensorFlow [128] to load the pre-trained CNN model for feature extraction, which is also provided. The model we use to estimate camera parameters and project coordinates from image space to world space and vice-versa is provided as a library by [125].

4.1.2 Parameters

The components of our system contain various parameters, as discussed below.

Object detection parameters The barrier state detector CNN is a TinyYOLOv3 network trained on our own data. The confidence threshold is 0.5 and the threshold used for non-maximum suppression is 0.3.

The person (and bicycle/dog) detector CNN is a YOLOv3 network pre-trained on MS-COCO. The confidence threshold for person detections is 0.5 and the threshold used for non-maximum suppression is 0.4.

Both networks use an image size of 416x416 pixels.

These settings have been chosen as an educated guess since estimating them systematically is outside the scope of this thesis.

Tracking parameters The Kalman filter in Deep SORT has parameters that control the covariance of observations relative to the model. A higher covariance in the observation matrix indicates more uncertainty in the observation, a higher covariance for the state indicates more uncertainty in the model. A balance between these is important, so that observations sufficiently influence the model while still mitigating the effect of outliers in the observations.

There are two weights that control the amount of uncertainty in the model, one for the position components and one for the velocity components of the observation matrix. By default, the weight for position is 1/20 and the weight for velocity is 1/160. While developing our implementation, increasing these weights to 1/15 and 1/120 to increase the uncertainty and relax the internal state's influence initially seemed to improve tracking somewhat, but after further testing on the entire data set this turned out not make any significant difference based on the metrics we use to evaluate tracking (see also Section 4.4).

The maximum distance for IoU matching is left at the default of 0.7, the maximum cosine distance for feature comparison is set to 0.5, the length of the feature gallery is set to 100, the maximum age of tracks before they are deleted is set to 50 frames (2 seconds) and the initialization period where successive detection to track matches are necessary before a track is confirmed is set to 10 frames (0.4 seconds). The latter helps to reduce false positive tracks by disregarding spurious detections.

Behavior analysis parameters The behavior analysis module has several parameters to control the delay for states to change. These values are set to the following, chosen as seemingly reasonable compromises between robustness and detection speed:

- Moving/not moving change: 0.75 s.
- Enter area: 0.5 s.
- Leave area: 1.0 s.
- Waiting for open barriers: 3.0 s.
- On tracks while barriers are closed: 0.5 s.
- Change rate for turn detection: 0.67 s.
- Alert cool down: 3.0 s.

There are also some threshold values. The speed threshold for movement is set to 0.35 m/s. As the determination of velocity in world space is somewhat inaccurate, this is chosen as a value that appears to work well enough. The running average of the velocity ($\bar{\mathbf{v}}$) is computed over the last $n_f = 25$ frames. The angle threshold in the turn detection that determines whether a change in direction was significant enough (θ_{max}) is set to $\frac{1}{2}\pi$. The number of frames considered in the moving average of the direction (n_{lb}) is set to 25. The distance a person needs to move after turning before the turn is considered a significant change in direction is set to 1.0 m. The minimum number of significant turns to be considered suspicious is set to 2, although when the barriers are open this is reduced to 1.

4.2 Ground truth annotation and training

To create training data for the barrier state detection network we developed a tool using Python and VLC [129] to manually annotate barriers in videos of railway crossings (see Figure 7). This tool allows a user to add events (moments where the state of the barrier changes) to the list on the right and create a set of bounding boxes associated with each event. These sets can be saved as presets and a preset can be quickly assigned to a an event by selecting it from the preset list below the event list, avoiding the need to recreate the bounding boxes each time. The event list can then be saved to a file and used in a script to generate bounding boxes for the barriers at any point in the video.

We annotated the barriers and their state with a bounding box in 60 videos of a railway crossing in a variety of different lighting conditions. We then used a script to extract 5000 randomly selected training images with annotations from these videos, which were used to train a small neural network to be able to detect the three different barrier states as objects. While these videos are of the same railway crossing, they are not the same videos as used in our evaluation data set.

An existing tool called CVAT (Computer Vision Annotation Tool) [130] was used to create ground truth annotations for person trajectories in the form a series of bounding boxes for each frame in the track. Together with these bounding boxes we annotated whether a person was moving, a cyclist and if he or she was exhibiting suspicious behavior.

We did not need to manually train a network for the person detector as persons, dogs and bicycles exist in the MS-COCO data set, which allowed us to use a pre-trained network. Feature extraction of persons in the tracker is provided by a pre-trained network provided by the authors of Deep SORT in their repository on GitHub.

4.3 Evaluation data set

To evaluate the performance of the tracker, we selected 50 video sequences containing a variety types of persons under different lighting and weather con-

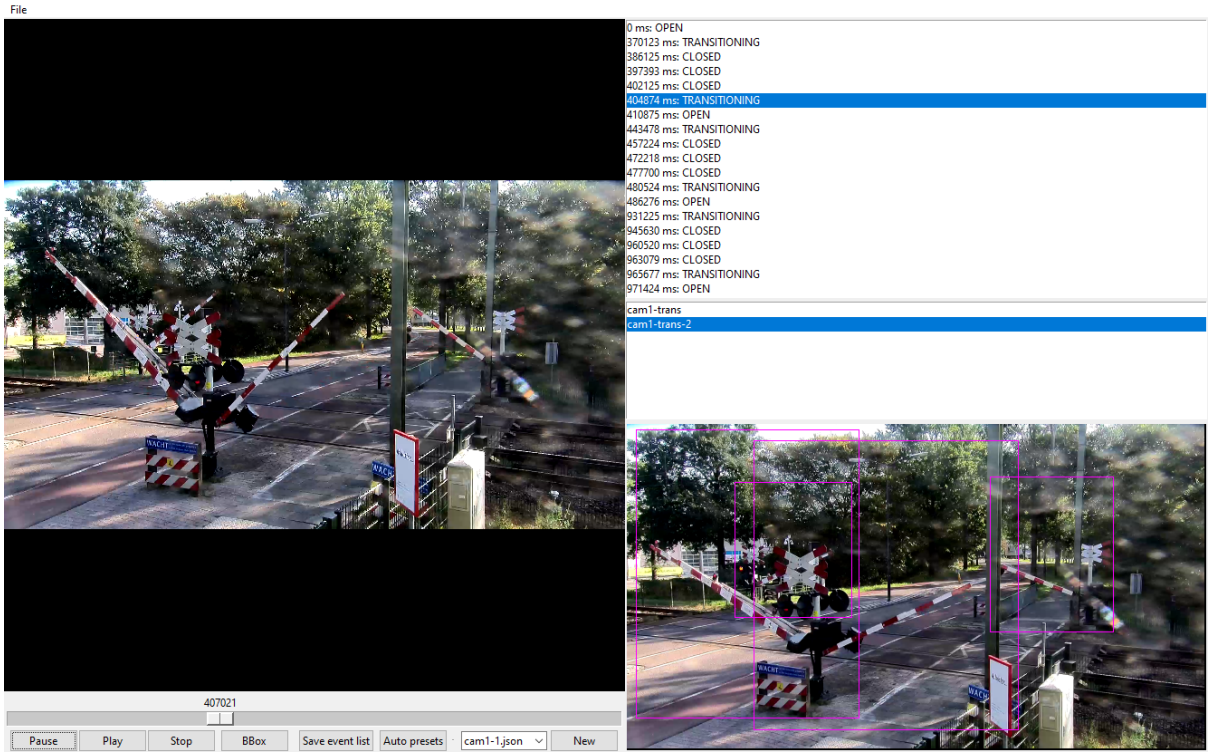


Figure 7: Barrier state annotation tool

ditions (see Figure 8 for examples) and annotated these with their trajectories. The set contains some easy as well as challenging scenarios for the tracker. In addition to that, we selected 6 video sequences that contained interesting behavior for qualitative analysis. See Table 1 for a listing of the videos and a short description of the events happening in each sequence.

4.4 Metrics

In this section we define metrics for evaluating the performance of the person detector, the person tracker and the behavior analysis.

4.4.1 Object detection metrics

At the basis of evaluating object detection algorithms lies intersection-over-union (IoU), which gives an indication of the amount of overlap between two bounding boxes.

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$

We use this metric to determine if two bounding boxes overlap more than some threshold, typically set at $IoU = 0.5$. Using the IoU metric and given the set of ground truth (GT) bounding boxes and detection bounding boxes, we determine:

- The true positive (TP) detections. A detection is true positive if it is a correctly detected ground truth box.
- The false positive (FP) detections. A detection is false positive if no ground truth box overlaps with the detection.
- The false negative (FN) detections. Undetected ground truth boxes are counted as false negatives.

Note that when multiple detections overlap the same GT box, we count the detection with the highest confidence as TP and the rest as FP.

From these we can compute the precision and recall values, which indicate the percentage of correct detections and the percentage of ground truths that were detected, respectively:

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

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

Precision and recall can be plotted against each other over varying confidence thresholds to gain insight in what settings work well. Increase the confidence threshold and the number of false positives (incorrect detections) will drop, at the cost of more false negatives (missed detections). Decrease the confidence threshold and the number of true positives (correct detections) will increase, at the cost of more false positives. Hence, a higher confidence threshold results in a higher precision, but lower recall. The precision-recall graph gives an indication

Figure 8: Some examples of the different lighting and weather conditions in the evaluation data set.

of what confidence threshold gives a good balance between the two.

The average precision (AP) is defined as a weighted average of the precision with the change in recall value as weight:

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

with P_n and R_n the precision and recall at the nth confidence threshold.

The mean average precision (mAP) is then defined as the average of the AP with $IoU \in \{0.5, 0.55, ..., 0.95\}$.

4.4.2 Object tracking metrics

We use the metrics proposed by [131] (summarized below) to evaluate the performance of our person tracker, with the thresholds $TR_{OV} = 15\%$ and $T_{OV} = 20\%$. For every video sequence, we determine:

- Correctly detected tracks (**CDT**). A ground truth (GT) track is considered correctly detected if the temporal overlap is larger than TR_{OV} and spatial overlap is larger than T_{OV}. Multiple system tracks can meet the conditions for one GT track.
- False alarm tracks (**FAT**). A system track does not have sufficient temporal overlap with any GT track, or it does not have sufficient spatial overlap with any GT track, despite having enough temporal overlap.
- Track detection failures (**TDF**). A GT track is considered not to have been detected if it does not have sufficient temporal overlap with any system track, or it does not have sufficient spatial overlap with any system track, despite having enough temporal overlap.
- Track fragmentation (**TF**) shows the lack of continuity of system track for a single GT track.
- ID changes (**IDC** shows the number of times a system track changes the associated GT track.
- Average track closeness for the whole sequence (**CTM**) and its standard deviation (**CTD**) is an indication of how much the pairs of associated GT tracks and system tracks spatially overlap for the duration of their temporal overlap.
- Average track matching error (**TMEMT**) and its standard deviation (**TMEMTD**) is a metric to measure the positional error of system tracks. It is an indication the distance errors between the centroids of an associated GT track and system track.
- Average track completeness (**TCM**) and its standard deviation (**TCD**) indicate the time span that a system track overlapped with a GT track divided by the total time span of the GT track.

Furthermore, we will identify and qualitatively describe several particular tracking failures.

4.4.3 Behavior metrics

While many aspects of the behaviors are difficult to annotate and quantitatively analyze, we can compare the movement state and start/stop events of ground truth tracks (GT) with that of associated system tracks (ST). We define 3 metrics:

1. Precision and recall of start/stop events, measured in TP, FP and FN. This gives an indication of how much of the events are detected within a given margin (time window).

2. Error of matched events (i.e. the time difference between the GT and ST event). This gives an indication of the lag between a start/stop event happening and the event being detected.

3. Closeness of the movement state. While the above metrics give an indication of how well the changes are detected at the right time, this metric gives an indication of how well the detected movement state correlates with that of the GT track regardless of matched events.

Additionally, we will qualitatively analyze what aspects of the behavior analysis and classification

work well, what aspects do not and illustrate these with examples.

Matching tracks ST tracks are matched to GT tracks, similarly to how we associate tracks for the object tracking analysis. An ST track overlaps with a GT track if it overlaps more than $TR_{OV} = 15\%$ in time and more than $T_{OV} = 20\%$ in space. The normalized temporal overlap is determined by dividing the length of the intersection of the two tracks in time by the length of the union of the two tracks in time. The average spatial overlap is determined by taking the average IoU of the bounding boxes of the two tracks for the length of time that they intersect temporally. We then sort these by the product of the normalized temporal overlap and the average spatial overlap, so that the ST track that most closely matches the GT track will always be considered first.

Precision and recall To gain insight in how much of the start/stop events are correctly detected, we try to match ST events to GT events. For each GT event, we search the sorted list of associated ST tracks for matches. To allow for a margin of error between the time t_{GT} of the annotated event and the time t_{ST} the event is detected we employ a threshold T_e to define a time window after t_{GT}, as well as an additional threshold T_s to account for inaccuracies in the time of the annotated event. An ST event is considered correctly detected if $-T_s \leq t_{ST} - t_{GT} \leq T_e$. Only the first event that satisfies this condition is considered a match and counted as TP. Unmatched ST events are counted as FP and any GT event without matched ST event is considered FN. Precision is then computed as $TP/(TP + FP)$ and recall as $TP/(TP + FN)$.

Error The amount of time between matched GT and ST events is expressed in the error metric. It is computed by averaging the errors of all events in the entire data set. Ideally, the error is zero.

Movement closeness The behavior analysis module keeps track of the state of movement at each frame. Although start/stop events might not be correctly detected within the thresholds, this does not mean the movement isn't largely correctly detected on a frame-by-frame basis, so it can give a skewed idea of the performance. Therefore, we use this metric as well to give an indication of how much of the GT track's movement state is correctly covered based on the state per frame. Since multiple tracks can be associated with a single GT track, the best matching track gets priority when filling in the state of the merged ST track. If the best matching track does not cover a certain frame, then the next best matching track is considered, and so on. Some annotated tracks contain segments that are occluded (for various lengths of time, from short to rather long), so we disregard these sections when computing how similar the two tracks are, which is done by diving the number of frames where the GT and ST states are equal by the length of time the GT track is unoccluded. The average closeness of a video sequence is computed by a weighted average of the closeness of each GT track using the length of the GT track as weights. The average closeness of the whole data set is computed the same way, by a weighted average of each track's closeness using the length of the track as weight. While computing the closeness using the raw movement state as logged by the behavior analysis module results in a single value that is invariant to the threshold used for associating events, we can also evaluate the closeness when using only the matched events, which does change with changing thresholds.

4.5 Evaluation

To evaluate the performance of our system we use the metrics defined in Section 4.4 for a quantitative analysis of detection, tracking and the movement detection component of the behavior analysis module.

We qualitatively evaluate the performance of the behavior analysis module by highlighting correctly and incorrectly detected events.

The results of this evaluation are presented and discussed in the next section.

5 Results and discussion

In this section we present the results of our experiments, evaluate them and discuss their implications.

Our implementation has not been designed with speed in mind, as it is only a proof of concept of functionality. However, it does not rely on batch processing and would theoretically be able to run in real time if the implementations of the subsystems are optimized well enough. Therefore, we will focus on the quality of behavior detection, disregarding processing time.

5.1 Environmental state detection performance

Since the barrier state detector is trained to detect only 3 different states of barriers as objects, with data from the same crossing as our evaluation set, the detections are of high quality and almost always correct. Combined with the smoothed transitioning the mechanism works almost without fail on our evaluation sequences. The only situation where it fails is when the barriers are initially in a transitioning phase, at which point the processed state defaults

to *opening*, even though it might actually be closing. The detector only detects a barrier state as open, closed or transitioning and a previous open/closed state is necessary to determine whether it's opening or closing. When the barriers finish the transitioning phase the state is changed to the correct state (i.e. open or closed). A more elaborate way of detecting the barriers would be necessary to correctly handle this situation (alternatively, it could be marked as undetermined), although it does not seem very consequential to the resulting behavior detection.

One drawback is that the network was not trained to also detect the state of the lights, as this turned out to be a useful piece of additional information (see Section 6.7).

5.2 Object detection performance

Speed One of the bottlenecks on our implementation is object detection, taking up a large percentage of the processing time for each frame. During development we circumvented this by caching the detection results, after which the rest of our system ran at speeds approaching real-time, despite algorithmic and implementation inefficiencies.

Since YOLOv3 has been reported to run at around 30 frames per second achieving a mean average precision (mAP) of 30 on the COCO data set, it stands to reason that by itself, given the right optimizations, YOLOv3 is capable of running in real-time. At the time of writing, YOLOv3 has been superseded by YOLOv4 [132], which achieves reported speeds of around 80 frames per second and achieving an mAP of 43 on the same data set. This puts it well within the realm of real-time object detection using a conventional GPU.

Accuracy The precision and recall of our complete data set is computed by determining the combined TP, FP and FN for all frames in all 50 video sequences by comparing the output of our person detector with the annotated bounding boxes for each frame. At $IoU = 0.5$ and a confidence threshold of 0.5, there are 21683 TP, 4175 FP and 10533 FN detections. Hence, for our person detector, $Precision = 0.839$ and $Recall = 0.673$.

The precision-recall curve can be found in Figure 9.

$AP = 0.679$ for $IoU = 0.5$ and $mAP = 0.380$ for $IoU \in \{0.5, 0.55, ..., 0.95\}$. For an overview of the precision-recall curves at all these IoU thresholds see Figure 21.

Occlusions One issue we identified with the YOLOv3 detector in particular is the way it presents bounding boxes. It does not handle (partial) occlusions very well and will give a bounding box around only the detected parts of an object. This means

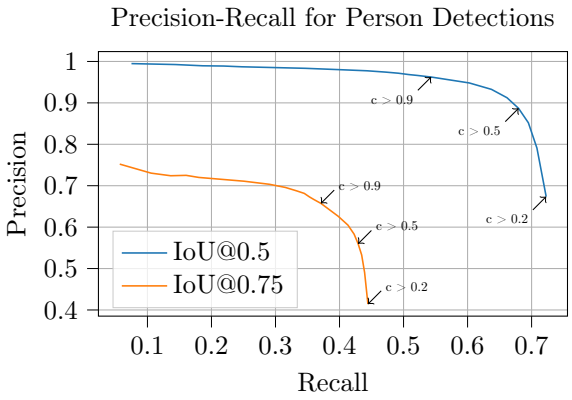


Figure 9: Precision-Recall curve of our YOLOv3-based person detector over different confidence thresholds. Computed at two different IoU thresholds for matching detections to GT bounding boxes.

the bounding box of a person will increasingly inaccurately represent the person before disappearing. This causes tracking problems, as explained in Section 5.3.

Image quality One area where the object detector seems to struggle is situations where visual fidelity is low. Extreme video compression artifacts due to fast motion, for example, can significantly reduce the image quality for some time (see Figure 10). Low contrast between persons and the background is another challenging situation for the detector (see Figure 16).

Figure 10: Video compression artifacts. Left: a train just passed by. Right: image quality stabilized after a second or two.

False positives As detection is not perfect, false positives are to be expected. The detector is only calibrated to detect persons, bicycles and dogs, but in some cases elements in the scene are incorrectly detected as such. In our evaluation data set this most often happens with parts of the barriers being marked as persons (see Figure 11). When these spurious detections are sporadic they do not present

much of a problem, as the tracker has measures in place to mitigate the influence of these spurious detections. However, in some cases these false positive detections persist for several subsequent frames, while the tracker has no way of knowing that these are false positive. See also Section 5.3.

Figure 11: False positive detection.

Bicycles and dogs Associating dogs and bicycles with persons works reasonably well in some scenes with fine-tuned multipliers for the scores. However, detection quality was too low for the mechanism to be useful. It is likely that YOLOv3 is not able to detect these objects reliably enough in our data set to be able to give consistent results (especially the dogs, which are small and often not clearly visible, making them hard to detect). Therefore, we have not considered the results of this mechanism further. Perhaps a different detector and/or one trained on a different data set might yield better results.

5.3 Person tracking performance

The performance of the tracker is analyzed by the metrics defined in Section 4.4.2. An overview of the results for all 50 annotated video sequences in our data set can be found in Table 2.

Of the 75 GT tracks in the data set, 72 were correctly detected. 2 tracks were not detected at all. There were 20 track fragmentations, indicating the amount of times a track is broken. Of the 121 detected tracks, 22 were considered to be a false alarm track. 29 ID changes occurred.

In short, most GT tracks are detected correctly, although their coverage is sometimes split over multiple partial ST tracks.

It is likely that tracker performance could be improved with systematic estimation of optimal tracker parameters, such as the Kalman filter weights, the maximum distance in the similarity metric, the maximum IoU distance and the maximum age of tracks. However, finding exactly the right combination of settings was too complex for the scope of this thesis.

Figure 12: One track fragmentation due to train passing by and several false alarm tracks (seq. 16).

False alarm tracks In several sequences more ST tracks than GT tracks are detected, such as in sequence 16 (see Figure 12). While some of these extra ST tracks are the result of multiple tracks covering a single GT track, for example because a track is broken partway through, and thus partially correct (these show up in the track fragmentation (TF) metric), some ST tracks cannot be matched to any GT track and are considered false alarm tracks (FAT). These are often caused by a period of consistent false positive detections. In this case there are several tracks initialized and activated on a part of the barriers that are being detected as a person. These tracks that cannot be associated with a GT track can potentially trigger false positive alerts.

Due to the initialization period of 10 frames and threshold of a minimum of 10 frames of track length for tracks to be considered in the behavior analysis and the evaluation, the number of FAT according to our metrics is relatively low compared to the number of false positive person detections. Although quite many very short tracks are briefly picked up by the tracker, as a track is initiated for every unmatched detection, this does not influence behavior detection greatly, as they are generally not confirmed and are quickly deleted as soon as a single detection is missed in the tentative period. If there happen to be enough subsequent detection matches for an incorrect track to be activated, it often does not exist long enough to trigger any kind of warning. However, there is a possibility that they do persist and do cause false events to be detected (see Figure 13).

The longer this initialization period is, the longer it takes for a real track to be activated. At the same time, a longer initialization period makes it less likely for false alarm tracks to occur, as there needs to be an increasing amount of subsequent false positive detections for it to be activated. Thus, there is a trade-off here.

Partial occlusions Nearly all persons crossing the tracks in either direction are at some point partially occluded by a vertical beam. In many cases the tracker has no problem with picking up the track when the person reappears. However, depending on the way the bounding box of the detection changes in the frames leading up to complete occlusion, it

(a) Seq. 4 (b) Seq. 15

Figure 13: False alarm tracks. FAT in (a) persists long enough to trigger an alert for waiting while barriers are open.

can also occur that a track is broken at the point of occlusion. See Figure 14.

(a) Seq. 1 (b) Seq. 6

(c) Seq. 11 (d) Seq. 13

Figure 14: Successful (a, b) and unsuccessful (c, d) handling of occluded part in track.

In some cases, partial occlusion involving multiple persons can cause tracks to be broken and even associated with wrong detections (Figure 15), making the remainder of the track completely inaccurate.

When tracks are lost because of partial occlusions, this is usually because YOLOv3 gives bounding boxes around the detected parts of a body, which shrink when a person is more and more occluded. Deep SORT incorporates this shrinking of the bounding box into the model of the motion of the track that is updated by the Kalman filter. When detections are then completely missed because occlusion becomes too great, the Kalman filter will predict highly inaccurate motion, making the IoU based matching in Deep SORT unlikely to expect a new detection for this track in (roughly) the correct place. The feature based matching, that takes precedence over IoU based matching, can compensate for this in many cases, but not always, as there is also a maximum distance between the expected position and a detection during the detection to track association process.

Missed tracks Figure 16 shows a situation where a person on a bike is waiting for a train to pass. The weather is sunny, casting shadows and shining light on a cobweb in front of the camera, making parts of the image more blurred. This combination of (locally) low contrast and lower image quality caused the detector to be unable to detect the person before she was already halfway across the tracks.

Lost tracks When a train passes in front of a person waiting on the other side of the tracks, the track is usually lost. A new track is initiated when the person is visible again. This could be mitigated by more powerful re-identification system (see Section 6.4) that can match the previously lost track with the newly initiated track. However, the impact of these lost tracks is not that significant, as the events we are interested in are not dependent on these gaps in detection being bridged.

Broken tracks due to other causes (such as an unoccluded track being lost due to detector or tracker failure) could be more problematic, as the previous turn detection information will be lost with the track. If a track is lost right after a person turned around but before the turn was detected, this turn will likely not be detected at all.

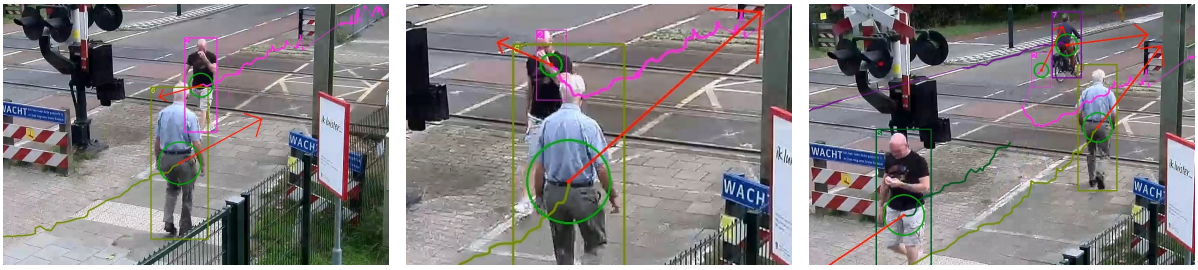
ID changes In several sequences one or more ID changes occurred. For example, in sequence 8 (Figure 17a) a person on the pavement is occluded while a cyclist is passing by. The track belonging to the person on foot got matched to the cyclist incorrectly. In sequence 9 (Figure 17b) a group of people passed behind a beam, after which the tracker matched the reappearing persons to the wrong tracks, as the persons belonging to the tracks were still occluded.

5.4 Behavior detection performance

We quantitatively evaluate the performance of the movement detection based on the data set of annotated video segments. We then qualitatively evaluate the behavior detected in a selection of videos.

5.4.1 Movement

We use a threshold of $T_s = 1$ s to allow association of ST events up to 1 second before the GT event to compensate for annotation inaccuracies. The T_e threshold defining the window after the GT event is varied from 0.5 s to 5 s. We achieve an average movement closeness of 0.777 on our data set when computed using the raw movement state from the behavior analyzer, with a standard deviation of 0.260. For precision, recall and average error of matched events, and average closeness computed using only matched events, see Figure 22.



(a) Two persons about to pass each other.

(b) Bounding box of one person shrinks due to partial occlusion. Detections are missed.

(c) Tracker is unable to keep the track. Associates it incorrectly with passing cyclist for a few frames.

Figure 15: Lost track due to partial occlusion making the detected bounding box smaller, causing the Kalman filter to predict highly inaccurate motion. (seq. 21).

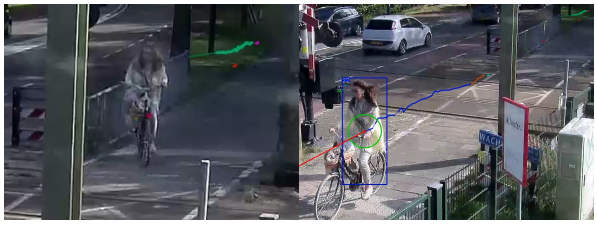


Figure 16: Person being in shadowy area and image quality being low due to illuminated cobweb in front of camera cause the detector to fail, making the tracker unable to pick up the person correctly until she becomes more visible (seq. 42).

(a) Seq. 8. (b) Seq. 9.

Figure 17: ID changes

The average closeness based on the raw state is higher than that when using only the matched events, although it starts to come close with larger thresholds. Note that using only the matched events to compute closeness can give a skewed result (see Figure 23), as it potentially ignores many imperfections in the movement detection. Hence, using the raw movement state gives a more accurate representation of the closeness of movement detection.

Examining the data, it appears that it often takes a while for the first movement event to be detected, sometimes quite long after the GT movement event (see also Figure 18). This can be explained by the compounded delays caused by the time it takes for the track to be activated by the Deep SORT tracker and then the time it takes for the behavior analyzer to determine the movement state. Missed first events can be exacerbated when a person is initially standing still, as this is the initial movement state of a new track, which means no state change event will be generated for it.

When a track is already established it seems that most movement change events are detected within a second or so (see also the average error), which is in line with expectations, as the delay to change movement state and trigger the corresponding event is set to less than a second. From the recall and precision graphs it would seem that after 2 seconds there are not that many more events that are correctly identified. This is also reflective of most events that do have a match being within 2 seconds, whereas beyond the 2 second threshold most of the extra correctly detected events are the first movement events of a track.

In practice, these missed startup events are not very important, as we are not really interested in detecting whether a person moves right after he or she enters the frame, but rather what a person is doing over time, when the track already had time to establish.

Closeness is further brought down by track fragmentations. Although frames where the GT track is occluded are not counted (and a lost ST track thus has no influence), when the person does reappear, there will be a gap caused by the initialization of the new track and the detection of movement (that is initially set to "not moving").

It appears that movement detection works adequately, although it would benefit from better object detection and tracking, as it would result in less lost tracks and thus more accurate movement tracking.

5.4.2 Qualitative analysis of behavior detection

In this section we evaluate a number of videos containing noteworthy events (some from the annotated set, although behavior events are not annotated) by presenting a graphical overview of all the events

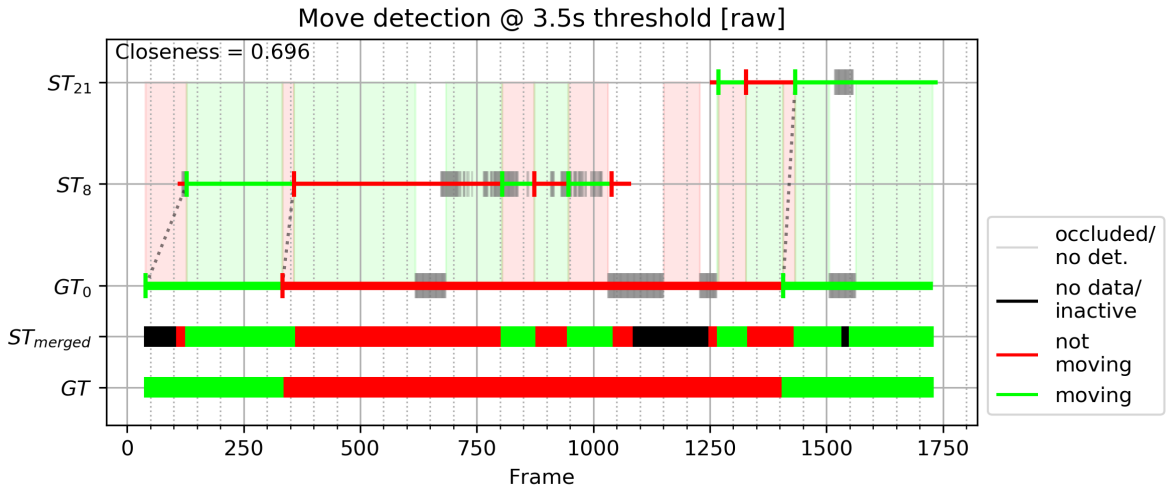


Figure 18: Illustration of movement detection evaluation. The shaded areas indicate where the merged ST track is equal the the GT track. Only these segments are used for computing the closeness. Matched events are connected by a dotted line.

being detected and/or a detailed depiction of the scene through screenshots.

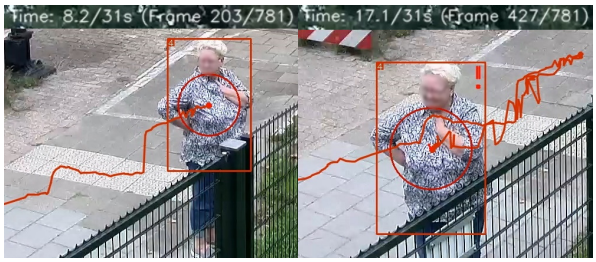


Figure 19: Sequence 9.

Sequence 9 (Figures 19 and 24) shows a person standing still next to the fence. 3 seconds after the person stops the *waiting for open barriers* event is detected, which triggers an alert. Just after the 12 seconds mark the person turns around and moves very slightly, enough for a turn to be detected. However, she has not yet moved enough to pass the distance threshold to trigger the *direction change* event. The position of this turn is remembered though, and the turn remains "queued", so that when she does move far enough from the point of turning at around 26 seconds the *direction change* event is finally triggered.

In sequence 21 (Figures 15 and 25) two persons pass each other. The track of one of the two persons gets incorrectly matched to a a passing cyclist moving in the opposite direction, which causes a false *turned around with barriers open* event to trigger.

In sequence 33 (Figures 20 and 26) a person starts waiting because the warning lights are on, which is the correct behavior. However, since our system is not currently able to detect the state of the warning lights and the barriers are still open, this triggers

Figure 20: Sequence 33.

the *waiting for open barriers* event, raising an alert. The system behaves as designed, but this is clearly not a desirable outcome. The person takes a few steps back just before a train arrives, which is detected as a *direction change* event at 38 seconds. However, this correctly does not trigger an alert, as the barriers are down.

The following sequences are the additional videos outside of the annotated data set. We will go through them and highlight interesting events.

In sequence 51 (Figures 27 and 30) a person walks up to the train tracks and stands still for a few seconds. About a second after the person appears to have stopped moving the *stopped moving* event is correctly detected. The person turns around and starts walking back, which is detected quickly and correctly. After the person has moved about 1 meter from where he turned around the *moving in different direction* event is detected and since the barriers were open, this also sends out a *turned around while barriers open* event, which triggers an alert. The person then starts moving backward around without stopping, which is detected about a second and a half later. After moving about 1 meter another *moving in different direction* event is

detected. The person stops at the tracks again and starts waiting a while. 3 seconds after the person stopped moving the *waiting for open barriers* event is detected, which does not trigger an alert, since the person was already in a "hot" state. The person turns around again and walks away.

All events in this sequence were correctly detected. By design, the *turned around* event is not detected when a person is standing still, so this event comes after the *started moving* event, although visually the person does turn around before moving.

In sequence 52 (Figure 31) a person starts waiting while the barriers are open, which is correctly and quickly detected, and raises an alert. Note that the entering and leaving of the track area is also correctly detected.

Sequence 53 (Figures 28 and 32) shows 2 persons going in opposite directions and crossing each other at the train tracks, where they stand still for a while before continuing on their way. As one person is occluded by the other, a track is lost, since Deep SORT was unable to match a detection to the track at that point (possibly due to an absence of detections). The predicted bounding boxes drifted too far from the actual trajectory, which meant that Deep SORT considered the person not a match when the person did reappear. Although this track was lost, a new track was initiated upon reappearance and all events were correctly detected in a timely fashion. Alerts were triggered since the two persons both stood still while the barriers were open.

Sequence 54 (Figures 29 and 33) contains a police officer walking along the train tracks right into a forbidden area at around 52 seconds. This is detected almost immediately, only a few steps into the area.

Sequence 55 (Figure 34) depicts a night-time scene with a person waiting for open barriers, which is correctly detected.

In sequence 56 (Figure 35) a person walking a dog crosses the tracks. The dog is briefly incorrectly detected as a person, long enough for a track to be initiated for it. The person turns around, which is detected correctly, and walks up to the fence in a shadowy area where he stands still for a while. The size of the detections here fluctuates greatly, jumping between nearly the whole person and only his top half. This gives the movement detection a hard time, as the person's position (and thus velocity) is determined based on the centroid of the bounding box, which to the behavior analysis algorithm now appears to be moving around. This is an issue that could be solved by either better detections or more smoothing of the trajectories. Interestingly, due to these jittery bounding boxes, here we have a situation where the person was (incorrectly at the time) detected to have turned around, and only after that was he detected to have stopped moving. The bounding box centroid moved enough (more than 1 meter) from the point where the turn was detected, which raised the *moving in different direction* event, before the person was actually considered to be moving again.

From our evaluation set, it appears that the behaviors we attempt to detect are mostly detected correctly. It largely depends on the quality of the tracks, as this is all the information about a person that is available to the behavior analysis. Given decent tracking quality, the behavior detection appears to work quickly and accurately.

Broken tracks are not much of an issue, as long as they remain relatively accurate during the period they are active. It can occur that a track is incorrectly matched to a different person (resulting in an ID change), at which point highly inaccurate behavior might be detected.

6 Future work

There are many areas of our method where functionality could be improved or extended. We discuss some of these in this section.

6.1 Influence of detection quality on tracking

The quality of detections greatly influences the performance of the tracker (as described in [62]). Hence, it could be interesting to experiment with different detectors to see if they increase the detection rate.

As mentioned in Section 5.2, YOLOv3 is not aware of occlusions. There are other detectors available that handle occlusions much more gracefully and will infer the actual size and position of a person, even though the person might be partially obscured [41, 44]. Using one of these detectors instead would likely significantly enhance the performance of the tracker and result in less broken tracks. When detection quality rises, the need to filter out detections in difficult areas before sending them to the tracker subsides as well, allowing for greater coverage per camera.

Other ways to detect occlusions which can be used to extend existing detectors and trackers have also been proposed [42, 43].

A consideration is the performance of these algorithms, as one of the key factors making YOLOv3 attractive is its speed, whereas many other trackers with better detection quality, such as R-CNN based methods, are slower, often dipping below what can be deemed real-time.

6.2 Parameter optimization

There are many parameters in our system that influence the quality of object detections and object tracking. For the most part we have simply set these

to default values or to values that seemed to work. It is very likely that with a systematic approach to optimizing these parameters it is possible to increase the performance of the object detection and tracking algorithms we used. Especially the Deep SORT tracker has parameters that greatly influence how quickly tracks are picked up and deleted, how the bounding box position is predicted in the absence of detections, how much appearance influences matching and how far apart bounding boxes can be to be considered a match.

Additionally, the various parameters of the behavior analysis module can be tuned to taste, as real world situations might call for different values (such as the amount of time a person needs to stand still before it it considered waiting)

6.3 Training

Training the person detector with a more specific data set, e.g. based on footage from perspectives typical for railway crossing cameras, will likely yield better detection results (in this application) than those from the network that was trained on just MS-COCO, which is only a general data set. It is possible to fine-tune the existing network trained on MS-COCO with such a more specific data set using transfer learning.

While the barrier state detector works well for our evaluation set as it was trained on the same scene, if it were to be used on different scenes, it would need to be trained on a larger variety of railway crossings.

6.4 Tracker post-processing

If tracking quality is not sufficient, additional post-processing steps on the trajectories can be applied to get better results.

Smoothing trajectories Depending on the tracker and its parameters, jittery object detections can lead to jittery trajectories. Taking the original trajectory and applying a smoothing algorithm (e.g. some variant of a Kalman filter [133] or a moving average) can yield more realistic paths and less noisy data for the behavior analysis to work with.

We implemented a post-processing smoothing step over the tracks produced by Deep SORT using a Kalman filter, which did result in visually smoother trajectories, but found that it did not make a significant difference in the behavior analysis, so we deemed it redundant in our application, as it did not aid in the actual tracking process.

Re-identification Tracks can get broken due to occlusions. An additional re-identification step can be applied to match currently active tracks to previously lost tracks, tying them to the same person. Re-identification can be done by for example training a neural network to extract some set of features from a person and defining a metric to compare these features [64]. Deep SORT already employs this technique, but only in the short term, until the tracks reach their maximum age and are removed from the tracker. It can be useful to apply re-identification over longer periods of time, in order to keep track of persons walking in and out of the camera view, for example.

6.5 Handling PTZ cameras and tracking in world space

Pan-Tilt-Zoom cameras have different characteristics than regular static cameras. They come with a different set of challenges such as dealing with calibration, controlling PTZ parameters and estimating ego motion [134]. The current system does not take these into account.

Currently our tracking system works mostly in screen space without any additional depth information, which limits our ability to accurately track people. Since there are usually at least two PTZ cameras at each crossing, it is theoretically possible to combine them to build a world space representation of the crossing and track people in three dimensions, which would allow for more accurate tracking, as well as being more robust to occlusions.

This way, more detailed information about persons relative to the scene would available, which should improve the chances of correctly detection suspicious behavior. For example, it will be easier to determine on which side of the barriers a person is, or if there is a person hiding behind a raised barrier, waiting for a train to arrive. Furthermore, the velocity of persons, while already measured in a rough approximation of the world space, can be more accurately measured with depth information and when the tracking is also done in world space, improving the potential quality of the behavior detection.

In [135] a method is proposed to extract camera calibration directly from the visual content of the frames continuously and in real time, which is then used to track targets in 3D world space. This method could replace the Deep SORT tracker so our behavior analysis method becomes invariant to camera motion.

6.6 Detecting other cues

Given the widely successful application of deep learning in computer vision in recent years, it is likely possible to apply a neural network to directly detect a multitude of different cues that indicate strange behavior from video data. One of the main chal-

lenges in this is acquiring a large enough data set with corresponding annotations of the cues to learn. Types of cues one can think of are the act of smoking a cigarette, types of clothing, gait and posture, facial expression, etc. Should a system be available that can detect these types of cues, it could be integrated in the behavior analysis module to bring more nuance to the classification.

Pose estimation Going beyond simple object detection, approaches such as [27, 46, 47] allow for estimation of the pose of detected persons, which could be included in the behavior analysis to gain a more accurate idea of what a person is doing. For example, a person sitting down can be considered suspicious, which could be detected as such with this approach.

6.7 Environmental state detector

The module that detects the state of the barriers can be extended to give more detailed information about the scene. For example, it would be useful to train it to also detect whether the warning lights are on. One practical use of this is the situation where a person decides to start waiting as soon as the warning lights come on but before the barriers come down. This is expected behavior, but our system is currently still likely to classify this as suspicious since the person is technically waiting for open barriers.

6.8 Areas of interest

In the current implementation, the areas that are of interest such as the train tracks and its forbidden zones are annotated manually. In the real world this works for static camera positions, but it does require calibration up front. However, in the case of moving cameras, such as the PTZ cameras used by ProRail, it would likely be useful to train a neural network to detect these areas from any perspective and add this to the ESD module for added flexibility. That way the camera can change perspective without losing track of the correct areas. This problem falls in the class of semantic segmentation, where the pixels in an image belonging to the same object class are clustered together.

6.9 Other data sources

If other sources of data are available, such as the approach of trains or statistics about suicide attempts this could be used to add nuance to the behavior analysis and classification. For example, if suicide attempts are more prevalent at certain times of day according to the statistics, we could take this into account when deciding if certain behavior is suspicious, perhaps lowering some threshold for certain behaviors to trigger an alert whereas such behavior would otherwise not trigger an alert.

6.10 Machine learning-based behavior classification

Our rule-based behavior classification module could be replaced by a module based on machine learning. Currently we employ some hard rules to make a decision based on some combination of detected behaviors. These rules are a limiting factor in the decision making process, while the subject matter is highly subtle. Being able to discriminate within a vast set of subtle cues could be key in the usefulness of a system such as this. Hence, it would make sense to train a classifier on the behaviors to be able to take into account all different combinations of behaviors, incorporating more factors that might be negating or increasing the suspiciousness than what is feasible with hard rules. A traditional ML approach such as SVM could be applied here, but it would also make sense to train a neural network architecture suitable for learning temporal events.

When going that route, though, it could also be of interest to consider techniques such as [136], which attempt to learn to detect abnormal events in video data directly. Anomaly detection techniques could also be used, for example using the trajectories of persons and environment state as input.

6.11 Optimization and performance

There is room for many significant optimizations in our software platform. Application of suitable data structures for lookups will make the runtime of the behavior analysis negligible, as it is not a computationally expensive process algorithmically, and the amount of tracks in a real world system would remain relatively low since there are not many active tracks at any given time (and old tracks can be deleted).

Hardware acceleration exists for many deep learning tasks. For example, MXNet and Intel OpenVINO allow for optimized neural networks to run at very high speeds. OpenCV can be built with support for the OpenVINO backend as well, to allow inference to be run on a GPU and other devices aimed at parallel computation tasks. The bottlenecks in our system (namely the object detector networks and the feature extraction in Deep SORT) that currently run using unoptimized implementations on the CPU can be replaced with these optimized implementations, which would make them orders or magnitude faster.

7 Conclusion

In this thesis we presented an approach for real-time suspicious behavior detection at railway crossings by employing a tracking by detection framework in combination with a single-stage CNN based object detector to obtain screen space person trajectories which are then projected into world space using a calibrated camera model. We introduced a method to extract basic motion- and location-based information about a person's behavior from these world space trajectories, that together with the detected state of the environment is used to classify a person's behavior as being suspicious or not. Events are sent out when behavior state changes and alerts are triggered when specific conditions are met.

In addition to presenting the general design of our approach, we implemented and evaluated a proof of concept to investigate person detection, person tracking and behavior detection quality using a combination of YOLOv3 and Deep SORT. We found that tracking people using this combination works reasonably well, although better detections would benefit the tracking quality and the tracker would benefit from some kind of occlusion detection system. These techniques are relatively lightweight and have no problems running in real-time, provided an adequately optimized implementation that utilizes hardware acceleration is used.

Given the continued advancements in object detection and deep learning in general (see for example the gains YOLOv4 [132] made over YOLOv3), person detection should likely not be considered a limiting factor in our system. Multiple object tracking remains challenging, and while Deep SORT produces usable results with a very efficient approach, there are still significant gains to be made in tracking quality.

The behavior analysis and classification system adequately detects the behaviors listed in Section 3.2 within a reasonable amount of time and with only a few errors, some of which can be corrected without much extra work. Other errors are strictly the result of the tracker making mistakes, such as swapping identities between persons. The method itself is not computationally expensive and run time is negligible relative to the much more compute intensive tasks of object detection and tracking.

In conclusion, our system can detect several types cues for suspicious behavior based on the trajectories of persons. Highly performant methods are available for all the components that constitute our architecture, making it possible for such a system to run in real-time, thus making it suitable for surveillance applications.

While still relatively basic, we hope that our method can help prevent suicides at railway crossings and provide a basis for more accurate and nuanced behavior detection in the future.

A Algorithms

A.1 Behavior analysis

Algorithm 1: Turn Detection

def `UpdateTurnDetection`$(t, \mathsf{v}, \mathsf{moving}, \theta_{max}, n_{lb}, \delta, dt, \mathbf{d}_c, n_c, s)$**:**

  Data: t = current frame index
  $\mathsf{v} = \{\bar{\mathbf{v}}_0 \dots \bar{\mathbf{v}}_t\}$
  $\mathsf{moving} = \{is\_moving_0 \dots is\_moving_t\}$
  θ_{max} = angle threshold
  n_{lb} = number of frames to look back for moving average
  δ = delay time in seconds
  dt = time since last update
  $\mathbf{d}_c$ = cumulative direction
  n_c = number of cumulative direction samples
  s = turn score
  Result: Updated internal state variables $\mathbf{d}_c$, n_c and s, Turn Detection Event

  begin
    $turn\_detected \leftarrow False$
    if $\mathsf{moving}[t]$**:**
      `/* direction running average */`
      $\mathbf{d}_c \leftarrow \mathbf{d}_c + \mathsf{v}[t]/\|\mathsf{v}[t]\|$
      $n_c \leftarrow n_c + 1$
      $\bar{\mathbf{d}}_c \leftarrow \mathbf{d}_c/n_c$
      `/* direction moving average */`
      $\mathbf{d}_m \leftarrow (0, 0)$
      $n_m \leftarrow 0$
      $i \leftarrow t$
      while $i \geq \max(0, t - n_{lb})$ **and** $\mathsf{moving}[i]$**:**
        $\mathbf{d}_m \leftarrow \mathbf{d}_m + \mathsf{v}[i]/\|\mathsf{v}[i]\|$
        $n_m \leftarrow n_m + 1$
        $i \leftarrow i - 1$
      $\bar{\mathbf{d}}_m \leftarrow \mathbf{d}_m/n_m$
      `/* threshold angle */`
      if $\|\bar{\mathbf{d}}_c\| > 0$ **and** $\|\bar{\mathbf{d}}_m\| > 0$**:**
        $\hat{\mathbf{d}}_c \leftarrow \bar{\mathbf{d}}_c/\|\bar{\mathbf{d}}_c\|$
        $\hat{\mathbf{d}}_m \leftarrow \bar{\mathbf{d}}_m/\|\bar{\mathbf{d}}_m\|$
        $\theta \leftarrow \arccos(\text{clamp}(\hat{\mathbf{d}}_c \cdot \hat{\mathbf{d}}_m, -1, 1))$
        $turn\_detected \leftarrow \theta > \theta_{max}$
    `/* delay the turn detection */`
    $r \leftarrow dt/\delta$
    if $turn\_detected$**:**
      $s \leftarrow s + r$
    else:
      $s \leftarrow \max(0, s - r)$
    $turn \leftarrow$ NO_TURN_DETECTED
    `/* reset turn score and direction running average */`
    if $s \geq 1$**:**
      $s \leftarrow 0$
      $\mathbf{d}_c \leftarrow (0, 0)$
      $n_c \leftarrow 0$
      $turn \leftarrow$ TURN_DETECTED
    return $turn, \mathbf{d}_c, n_c, s$

B Additional results (tables)

B.1 Data set

Below is an overview of the evaluation data set.

#	Len	Time	Wthr/lghtng	Events	Notes
1	00:27	07:00	Medium contrast	Single person walking away from the camera.	
2	00:23	08:40	Medium contrast	Cyclist on pavement coming towards camera. Cyclist on road moving away from camera.	
3	00:11	08:40	Medium contrast	Cyclist on pavement moving away from camera	
4	00:50	09:20	Medium contrast	Single person walking away from camera.	
5	01:38	09:40	Medium / high contrast	Elderly person with rollator moving towards camera.	
6	00:43	11:40	Medium / high contrast	Single person walking towards camera.	Barriers closing at end.
7	01:06	12:20	Medium / high contrast	Cyclist on pavement moving toward camera, overtaking a single person moving toward camera that waits for closed barrier while train passes before he continues on his way.	
8	01:17	13:00	Medium / high contrast	Single person moving away from camera. Waits for closed barriers while train passes. Barriers close again when he is crossing the track.	
9	00:31	13:40	Medium contrast	Single person waiting in front of the fence. Group of 6 people passing her. First person walks back in the other direction.	Person standing still is suspicious.
10	00:34	14:20	Medium contrast	Single person walking toward the camera. Cyclist on the road moving away from camera.	
11	00:22	15:00	Medium contrast	Person initially waiting for barriers on other side of the road. Walks towards the camera after barriers open. Cyclist on road moving away from camera.	
12	00:33	15:00	Medium contrast	Single person walking toward camera.	
13	00:53	20:00	Low / medium contrast	Barriers initially closed. Persons walks up and waits on camera side before crossing when barriers open.	
14	00:32	21:00	Low / medium contrast	Single person carrying skateboard walking towards camera. Gets on skateboard at the end.	
15	00:34	21:20	Dark. Monochrome	Person moving away from camera. Barriers start closing as soon as person is on tracks.	
16	01:12	22:00	Dark. Monochrome.	Person with dog waits for closed barriers on other side. Train passes and person crosses tracks.	
17	00:20	07:40	Medium / high contrast	One person on pavement and one cyclist on road moving away from camera.	
18	00:22	08:20	Medium / high contrast	One person crossing tracks as barriers start closing. Moving away from camera.	
19	00:34	08:30	Medium / high contrast	Barriers initially closed. After barriers open, one person and two cyclists (on the road) cross tracks.	
20	00:23	09:40	Medium contrast	Person with dog moving toward camera. Cyclist moving away from camera.	
21	00:41	10:20	Medium / high contrast	Two persons moving in opposite directions, crossing each other. Cyclist moving away from camera. Person moving away from camera takes right turn after barriers.	
22	00:12	10:40	Medium / high contrast	Person running toward camera.	
23	00:22	11:00	Medium / high contrast	Person with cell phone moving toward camera.	
24	00:30	14:00	Bright. High contrast	Person with dog moving toward camera. Two persons on scooters moving away from camera on road.	Shadowy areas

#	Len	Time	Wthr/lghtng	Events	Notes
25	00:54	16:20	Bright. High contrast	Barriers initially closed. Person on other side walks up to barriers and waits behind pole (occluded). Moves toward camera when barriers open.	Shadowy areas
26	00:26	16:20	Bright. High contrast	Person walking toward camera with bicycle in hand	Shadowy areas
27	00:20	17:20	Bright. High contrast	Person walking toward camera. Barriers start closing as person is on tracks. Person starts running.	Shadowy areas
28	00:31	21:00	Dark. Monochrome	Person walking away from camera.	
29	00:31	22:20	Dark. Monochrome	Person walking toward camera.	
30	01:24	12:00	Rainy. Medium contrast.	Person waiting for barriers on camera side while train slowly passes. Crosses tracks after.	Water droplets on camera.
31	00:13	12:00	Rainy. Medium contrast.	Person on mobility scooter moves towards camera on pavement.	
32	00:34	12:20	Rainy. Medium contrast.	Person with umbrella moves away from camera. Cyclist on road moves away from camera.	
33	01:10	13:20	Medium contrast	Person on other side of tracks starts waiting when warning lights come on but before barriers start closing. Train passes. Person crosses tracks. Cyclist moves away from camera.	
34	00:58	14:00	Medium contrast	Person waits for closed barriers on camera side while train passes. Crosses when barriers open. Scooter with 2 persons moves away from camera.	
35	00:24	14:20	Medium contrast	Person in high-visibility clothing moving towards camera.	
36	00:55	15:40	Medium contrast	Barriers start closing as person walks up on other side and sits behind pole (occluded). Train passes. Person gets up and crosses tracks. Cyclist moves away from camera.	
37	00:23	17:00	Medium contrast	Cyclist moving away from camera. Person pushing baby carriage moves away from camera.	
38	00:19	17:00	Medium contrast	Person moving toward camera.	
39	00:16	17:40	Medium contrast	Person running away from camera.	
40	00:28	20:00	Low contrast	Person with 2 dogs moving away from camera and turning right after barriers	
41	00:37	10:00	Medium contrast	Person moving away from camera. Overlaps with person moving toward camera and pushing a stroller before turning right.	
42	00:41	10:40	High contrast	Cyclist on pavement approaches closed barriers from other side. Waits for train to pass and crosses tracks on bike.	Shadowy areas. Visible cobweb in front of camera.
43	00:26	11:00	High contrast	Person walking away from camera. Cyclist on road moving away from camera.	Shadowy areas. Cobweb in front of camera.
44	00:22	11:20	High contrast	Person with dog moving toward camera.	Shadowy areas
45	00:28	13:40	High contrast	Person moving toward camera.	Shadowy areas
46	00:31	14:20	High contrast	Person waiting for closed on camera side while train passes. Barriers open and person moves away from camera. Cyclist moves away from camera.	Shadowy areas. Train passes.
47	00:27	16:00	Very high contrast	Initially two cyclists halfway across the tracks moving away from the camera. Person with baby stroller moving toward camera. Cyclist moving away from camera. Scooter moving away from camera, overtakes cyclist.	Shadowy areas
48	00:18	18:20	High contrast	Person moving toward camera.	Shadowy areas

#	Len	Time	Wthr/lghtng	Events	Notes
49	00:29	18:40	High contrast	Person waiting for barriers to open. Moving away from camera. Starts running after crossing tracks.	
50	00:27	19:00	High contrast	Person coming from path on right parallel to tracks. Turns left and crosses tracks.	
51	00:25	13:40	Medium contrast	Person walks up to the tracks, looks along the tracks, walks back a couple of meters, then walks back to the tracks where he waits a little while before turning around again and going back the way he came.	Suspicious
52	00:41	07:40	Medium / high contrast	Person moving away from camera starts waiting for open barriers for a while before crossing the tracks.	Suspicious
53	00:52	14:40	High contrast	Two persons crossing each other at the tracks and talking to each other for a while while standing still on the tracks.	Shadowy areas. Suspicious
54	01:02	20:00	Low / medium contrast	Police on the scene. Two persons with dogs crossing the tracks side by side. Police officer walking along the train tracks into forbidden area.	Should trigger alert
55	00:52	23:00	Dark. Monochrome	Person walking up to the train tracks and waiting there for a while before crossing.	Suspicious
56	00:34	15:00	High contrast.	Person with dog coming from path behind tracks. Crosses tracks and moves back a bit, standing still next to the fence while looking out over the tracks.	Suspicious

Table 1: Summary of the video sequences in the evaluation data set, showing for each sequence the length of the video in seconds, the time of day the video takes place, the weather and lighting conditions, a short description of the events and some notes about the video if applicable. Entries 1-50 are the annotated evaluation set. Entries 51-56 are not annotated but contain interesting behaviors.

B.2 Tracker performance

Sequence	GT	ST	CDT	FAT	TDF	TF	IDC	CTM	CTD	TMEMT	TMEMTD	TCM	TCD
1	1	1	1	0	0	0	0	0.74	0.08	8.82	4.91	0.93	0.00
2	2	3	2	0	0	0	0	0.34	0.08	273.32	169.78	0.87	0.09
3	1	1	1	0	0	0	0	0.72	0.13	11.57	8.29	0.87	0.00
4	1	2	1	1	0	0	0	0.78	0.12	6.95	5.86	0.96	0.00
5	2	3	2	0	0	1	0	0.73	0.14	9.58	8.10	0.74	0.12
6	1	2	1	1	0	0	0	0.79	0.11	5.22	5.02	0.89	0.00
7	2	4	2	0	0	2	1	0.59	0.09	52.95	44.17	0.26	0.13
8	2	3	2	0	0	0	2	0.57	0.17	138.01	93.52	0.73	0.18
9	7	7	7	0	0	2	12	0.16	0.10	225.30	126.36	0.85	0.15
10	2	2	2	0	0	0	0	0.48	0.08	177.52	83.46	0.85	0.00
11	2	3	2	0	0	1	1	0.66	0.12	78.18	38.03	0.80	0.28
12	1	1	1	0	0	0	0	0.80	0.15	6.18	6.46	0.98	0.00
13	1	2	1	0	0	1	0	0.79	0.09	13.23	8.29	0.70	0.00
14	1	2	1	0	0	0	0	0.75	0.12	7.42	7.03	0.98	0.00
15	1	3	1	2	0	0	0	0.49	0.05	77.73	18.27	0.96	0.00
16	1	5	1	3	0	1	0	0.75	0.11	7.56	7.45	0.48	0.00
17	2	3	2	1	0	0	0	0.72	0.08	9.73	6.30	0.99	0.02
18	2	3	2	1	0	0	1	0.51	0.12	185.94	49.14	0.92	0.06
19	3	4	3	0	0	1	6	0.29	0.09	157.04	90.25	0.82	0.18
20	2	3	2	1	0	1	1	0.40	0.24	122.54	149.98	0.73	0.20
21	2	4	2	1	0	1	1	0.36	0.16	198.14	194.73	0.71	0.06
22	1	1	1	0	0	0	0	0.69	0.12	11.11	6.05	1.00	0.00
23	1	1	1	0	0	0	0	0.77	0.08	7.42	5.38	1.00	0.00
24	1	5	1	3	0	1	0	0.42	0.20	47.52	32.84	0.31	0.00
25	1	3	1	1	0	0	0	0.76	0.13	11.48	9.04	0.32	0.00
26	1	2	1	0	0	0	0	0.55	0.13	16.93	9.82	0.96	0.00
27	1	2	1	0	0	1	0	0.68	0.18	11.96	10.04	0.42	0.00
28	1	1	1	0	0	0	0	0.77	0.08	7.82	6.15	0.85	0.00
29	1	2	1	0	0	1	0	0.83	0.06	5.71	3.39	0.49	0.00
30	1	1	1	0	0	0	0	0.79	0.07	11.74	6.99	0.99	0.00
31	1	2	1	0	0	0	0	0.60	0.21	21.67	21.48	0.92	0.00
32	2	1	1	0	1	0	2	0.54	0.11	106.83	73.24	0.50	0.71
33	2	3	2	0	0	1	0	0.69	0.09	73.18	43.10	0.59	0.05
34	1	2	1	1	0	0	0	0.78	0.11	7.66	5.36	1.00	0.00
35	1	1	1	0	0	0	0	0.83	0.06	5.74	3.45	0.96	0.00
36	1	2	1	1	0	0	0	0.48	0.08	165.62	108.22	1.00	0.00
37	1	3	1	1	0	1	0	0.57	0.27	28.86	42.16	0.74	0.00
38	1	1	1	0	0	0	0	0.82	0.09	6.55	7.03	0.98	0.00
39	1	1	1	0	0	0	0	0.68	0.10	9.00	5.62	0.87	0.00
40	1	2	1	0	0	0	0	0.69	0.18	12.72	12.04	0.58	0.00
41	2	3	2	0	0	1	0	0.50	0.17	64.74	52.59	0.68	0.01
42	2	4	0	2	1	0	0	0.00	0.00	0.00	0.00	0.00	0.00
43	2	3	2	0	0	1	1	0.44	0.09	188.42	94.23	0.52	0.05
44	1	1	1	0	0	0	0	0.77	0.09	9.25	7.67	1.00	0.00
45	2	2	2	0	0	0	0	0.76	0.12	6.71	6.50	0.51	0.48
46	1	3	1	2	0	0	0	0.51	0.08	82.11	36.81	0.97	0.00
47	2	3	2	0	0	1	1	0.36	0.20	127.90	97.85	0.27	0.09
48	1	2	1	0	0	1	0	0.72	0.15	12.67	14.31	0.58	0.00
49	1	1	1	0	0	0	0	0.77	0.14	11.07	7.79	0.99	0.00
50	1	2	1	0	0	0	0	0.78	0.17	9.28	10.38	0.70	0.00

Table 2: Tracker performance, showing for each video sequence the number of ground truth tracks (GT), detected system tracks (ST), correctly detected tracks (CDT), false alarm tracks (FAT), track detection failures (TDF), track fragmentation (TF), ID changes (IDC), average track closeness in the sequence (CTM), track closeness deviation (CTD), track matching error of the sequence (TMEMT), track matching error deviation (TMEMTD), average track completeness (TCM) and track completeness deviation (TCD).

8 Additional results (figures)

8.1 Object detection

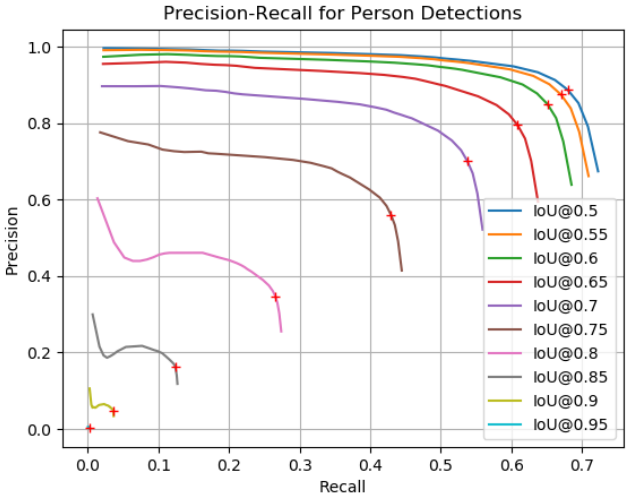


Figure 21: Precision-Recall curve of our YOLOv3-based person detector over different confidence thresholds at 10 different IoU thresholds. Markers indicate the point where the confidence threshold is $>$ 0.5. $AP_{IoU@0.5} = 0.679$ and $mAP = 0.380$.

8.2 Behavior analysis

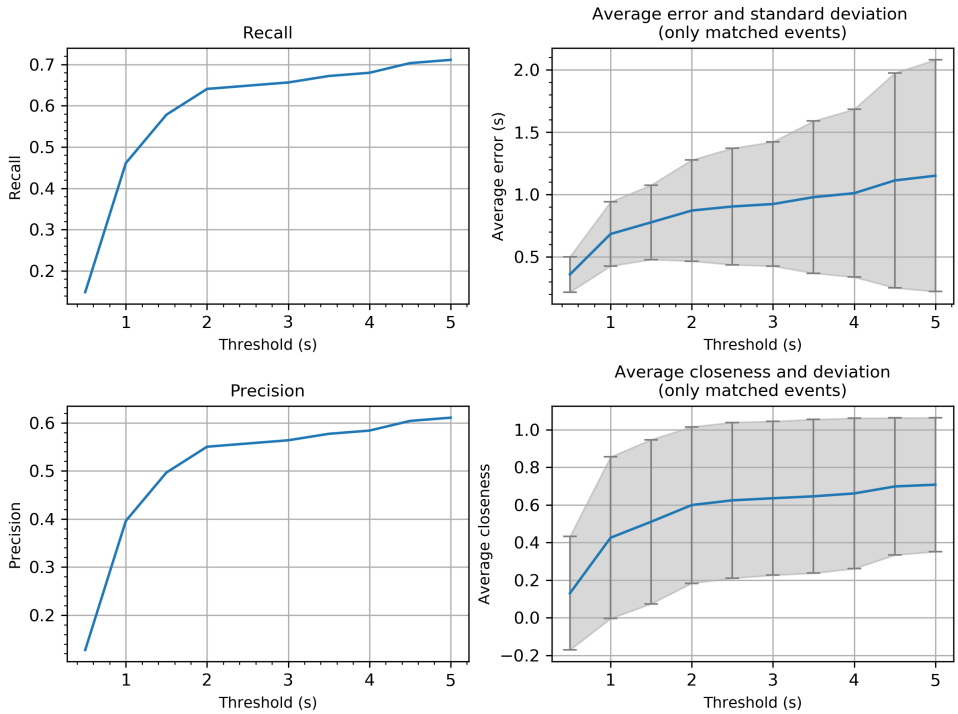
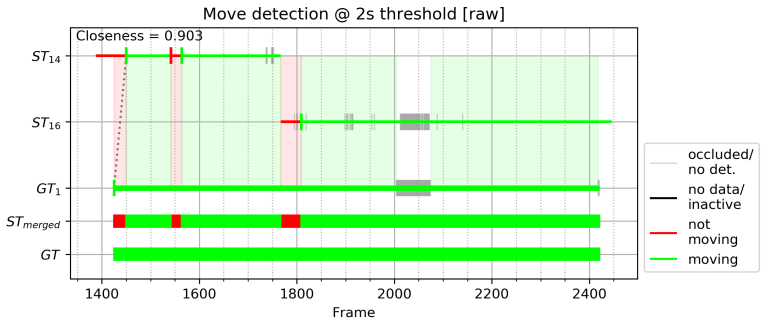
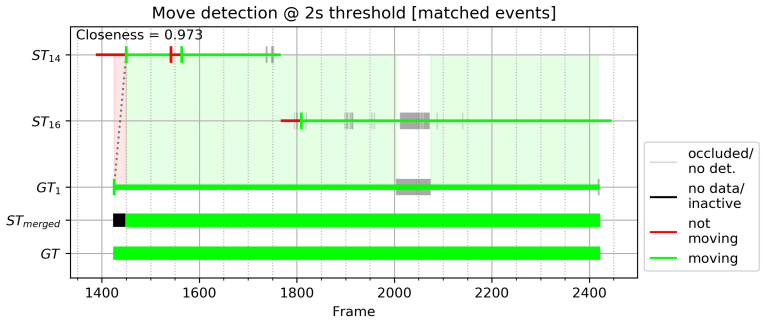


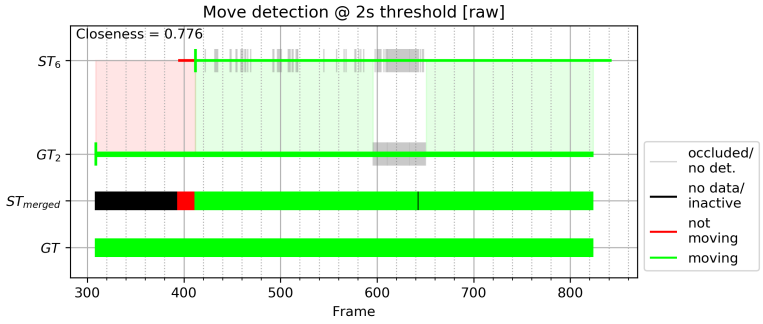
Figure 22: Recall, precision, average error and average closeness of movement of persons detected by the behavior analysis module. For reference: the average closeness based on the raw movement state is 0.777.



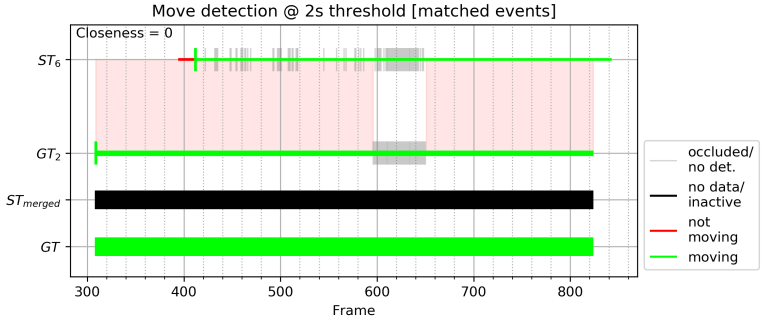
(a) Using raw states as logged by behavior analysis.



(b) Using only matched events gives higher closeness than using raw state, incorrect detections are unmatched and thus ignored.



(c) Using raw states as logged by behavior analysis.



(d) Using only matched events gives lower closeness than using raw state since there was no event matched at all, even though there is significant overlap.

Figure 23: Two examples of discrepancy between using raw state and matched start/stop events when computing closeness. Note that when using only matched events, it ignores the parts where movement is incorrectly detected, as well as parts where the track was lost and a new track was initiated. The shaded areas indicate whether the merged ST track is equal to the GT track or not at those frames.

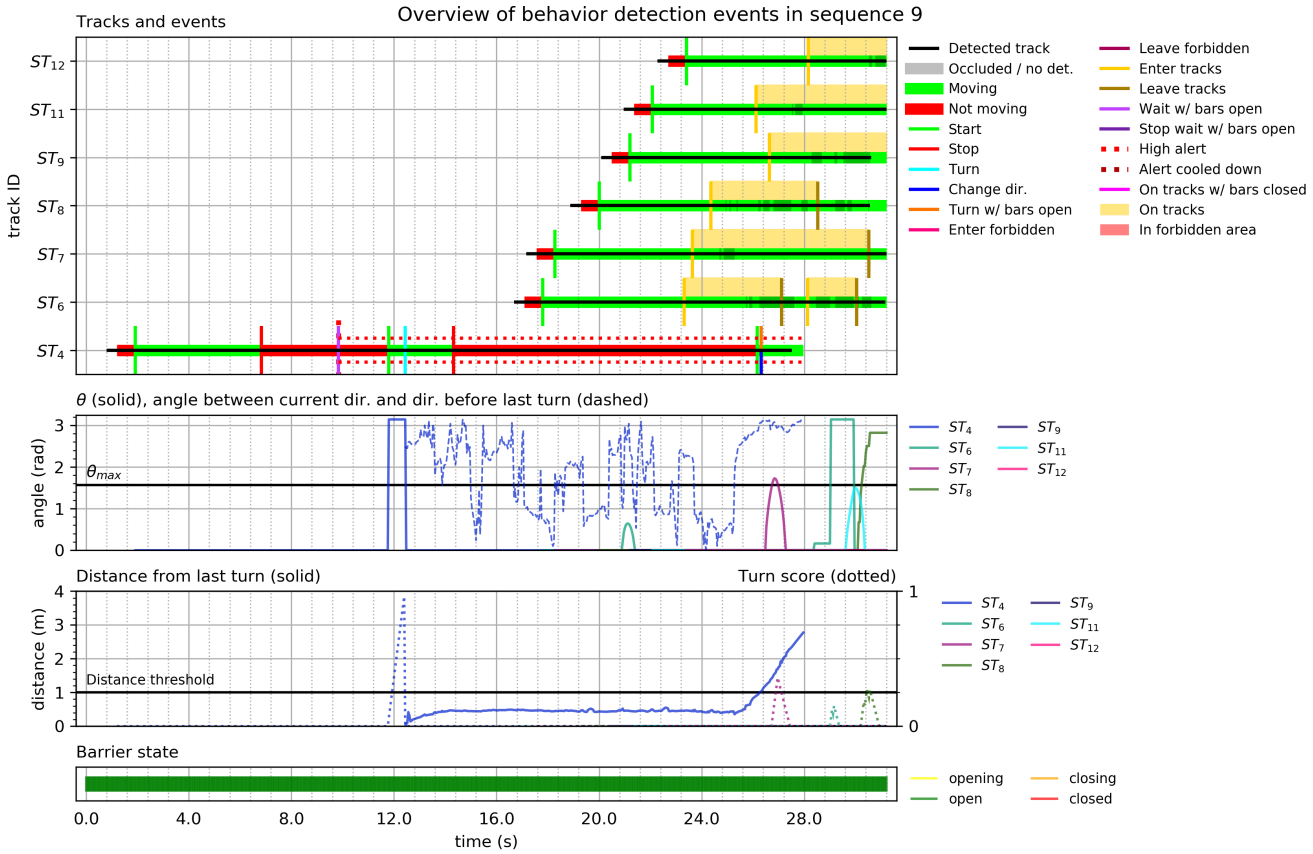


Figure 24: Events in sequence 9.

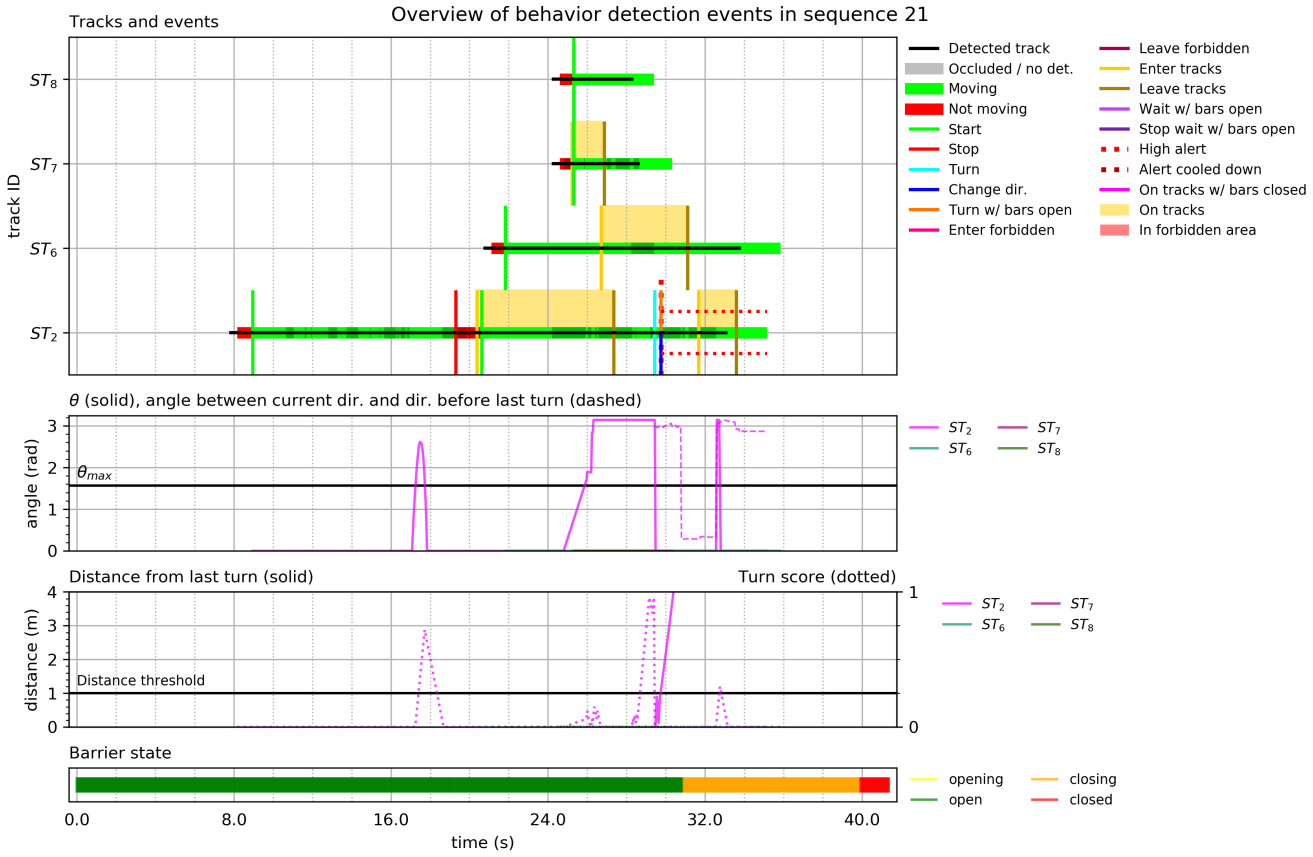


Figure 25: Events in sequence 21.

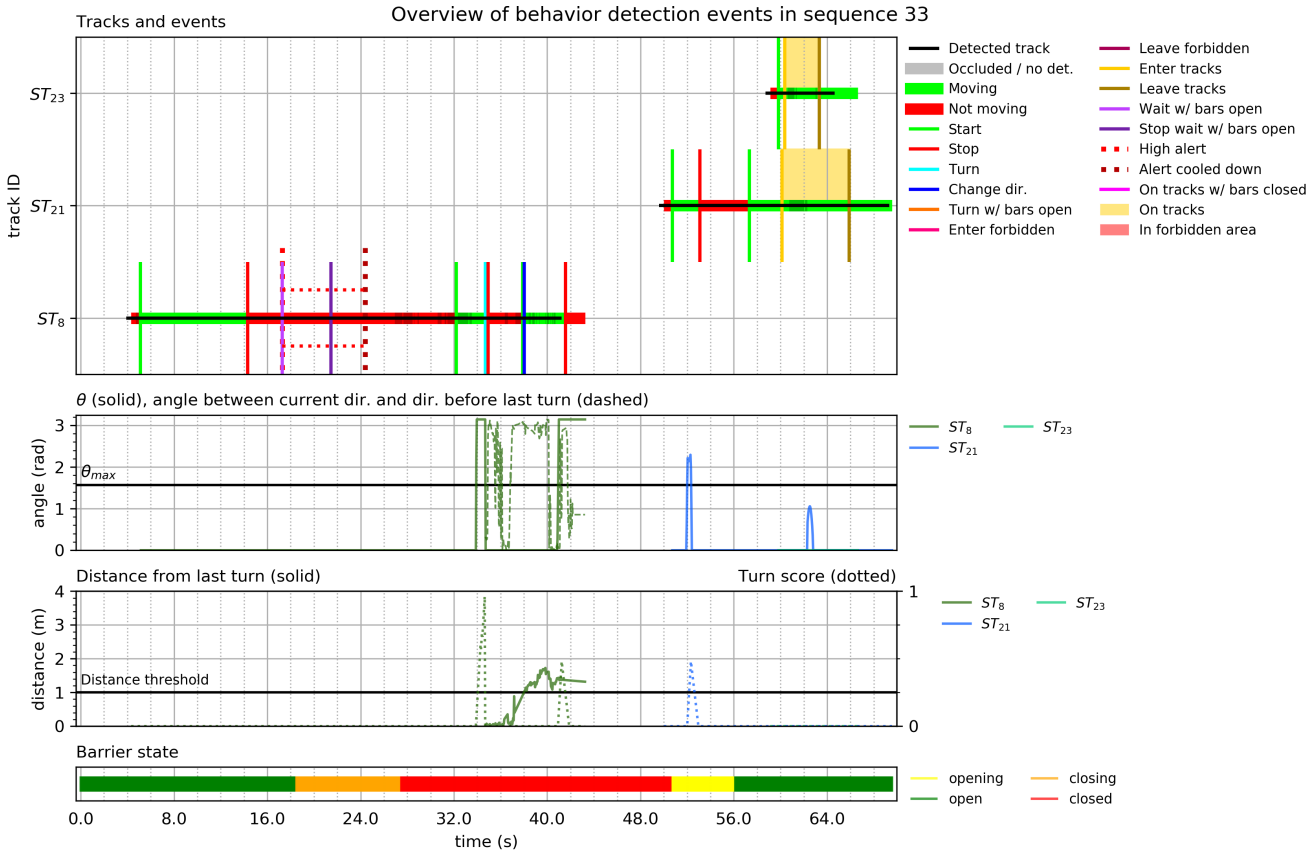


Figure 26: Events in sequence 33. A train passes at 40 s, occluding the person, hence the lost track.

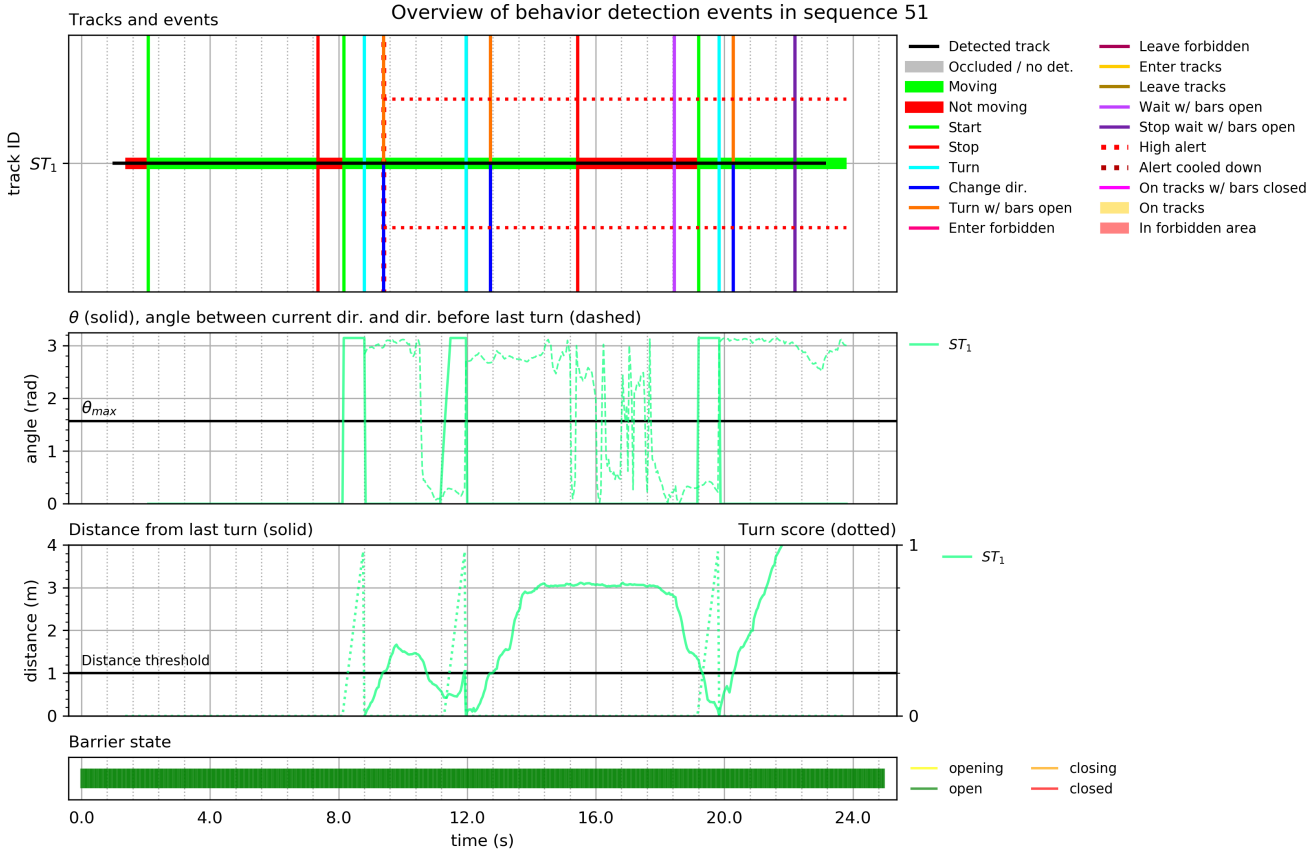


Figure 27: Events in sequence 51.

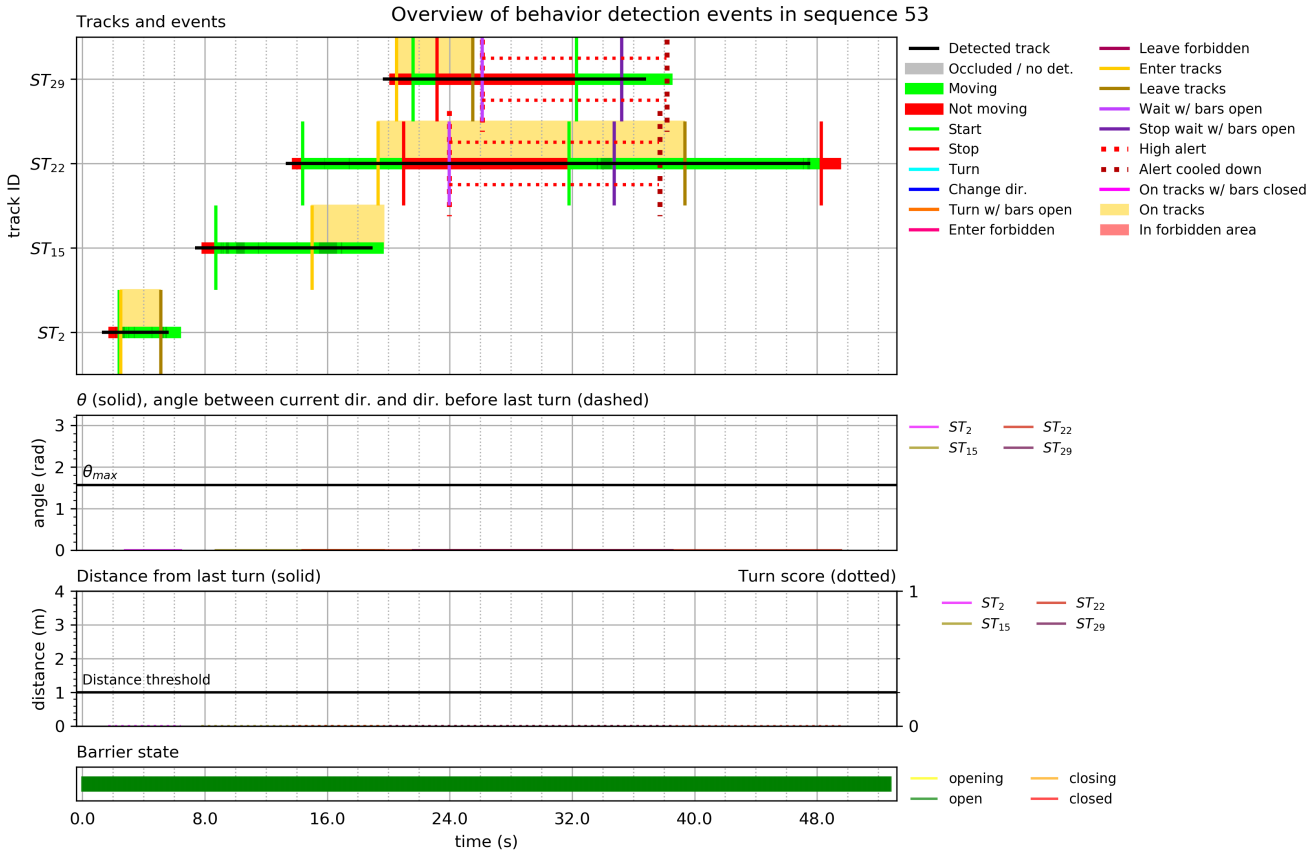


Figure 28: Events in sequence 53.

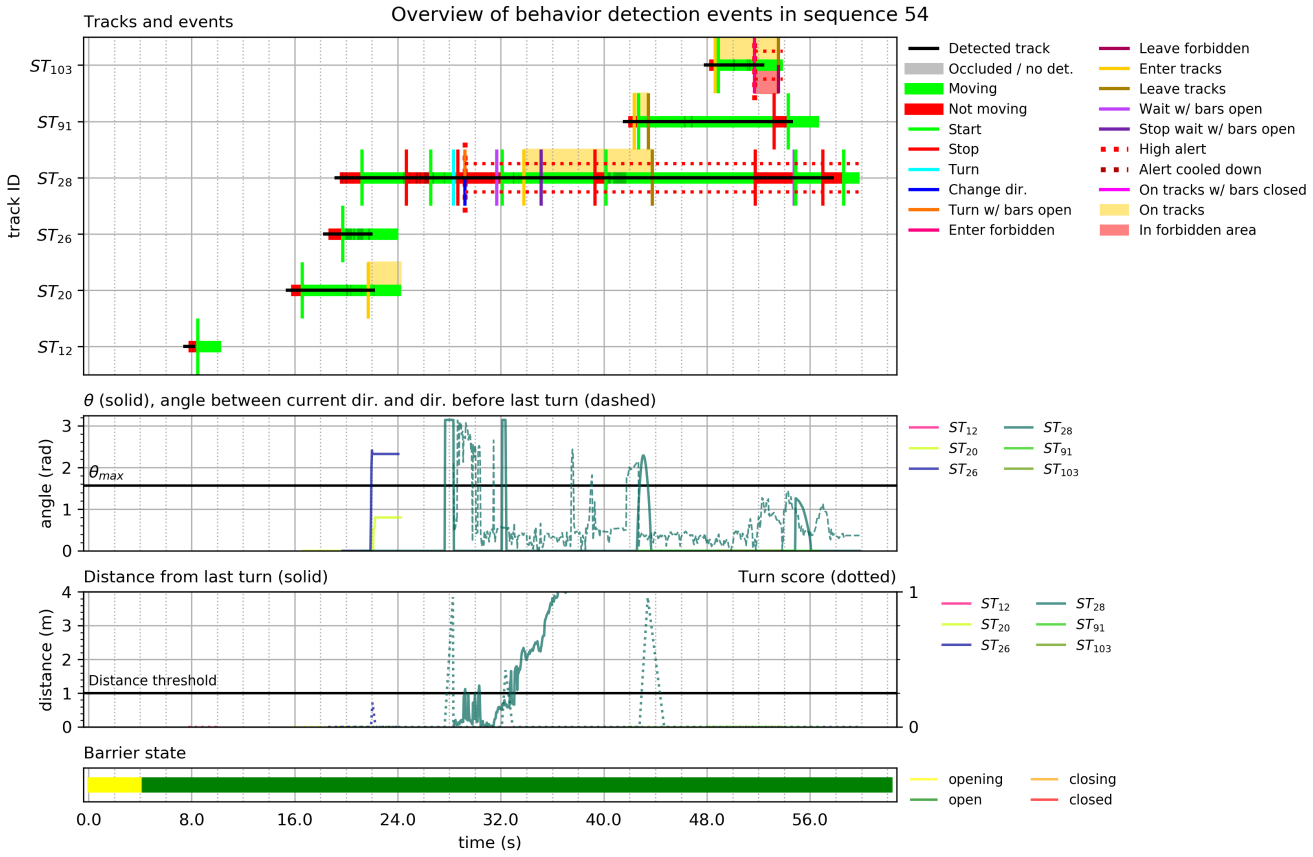
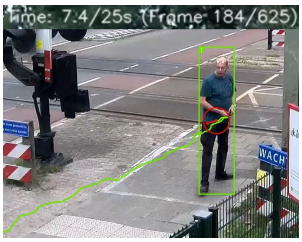


Figure 29: Events in sequence 54.

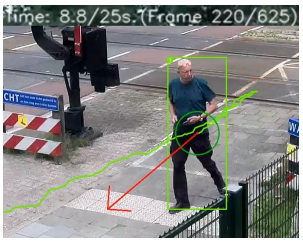
(a) Detected events with frame number.

(b) Person stops at tracks.

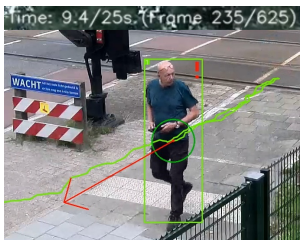


(c) After 1.2 s stop event is detected.

(d) Person turns around.

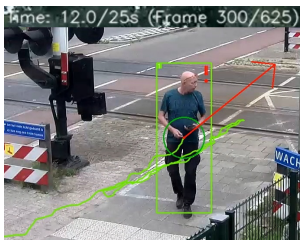


(e) In about a second the turn is detected.

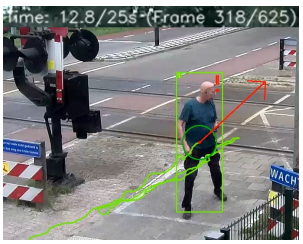


(f) Person moved significantly after direction change was detected. Barriers are open, so high alert event is triggered.

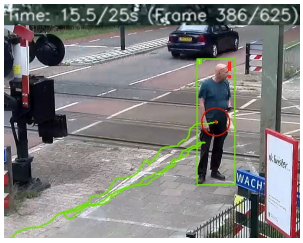
(g) Person quickly turns around without stopping.



(h) Turn is detected after 1.6 s.

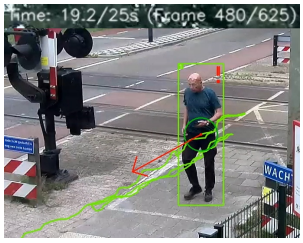


(i) Second significant direction change is detected.

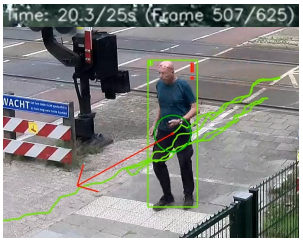


(j) Person stopped moving.

(k) After waiting a few seconds, *waiting for open barriers* event is detected, but cool down period for previous high alert has not expired, so another alert isn't triggered.



(l) Third turn detected.



(m) Third siginificant direction change detected.

Figure 30: Detailed description of events in sequence 51.

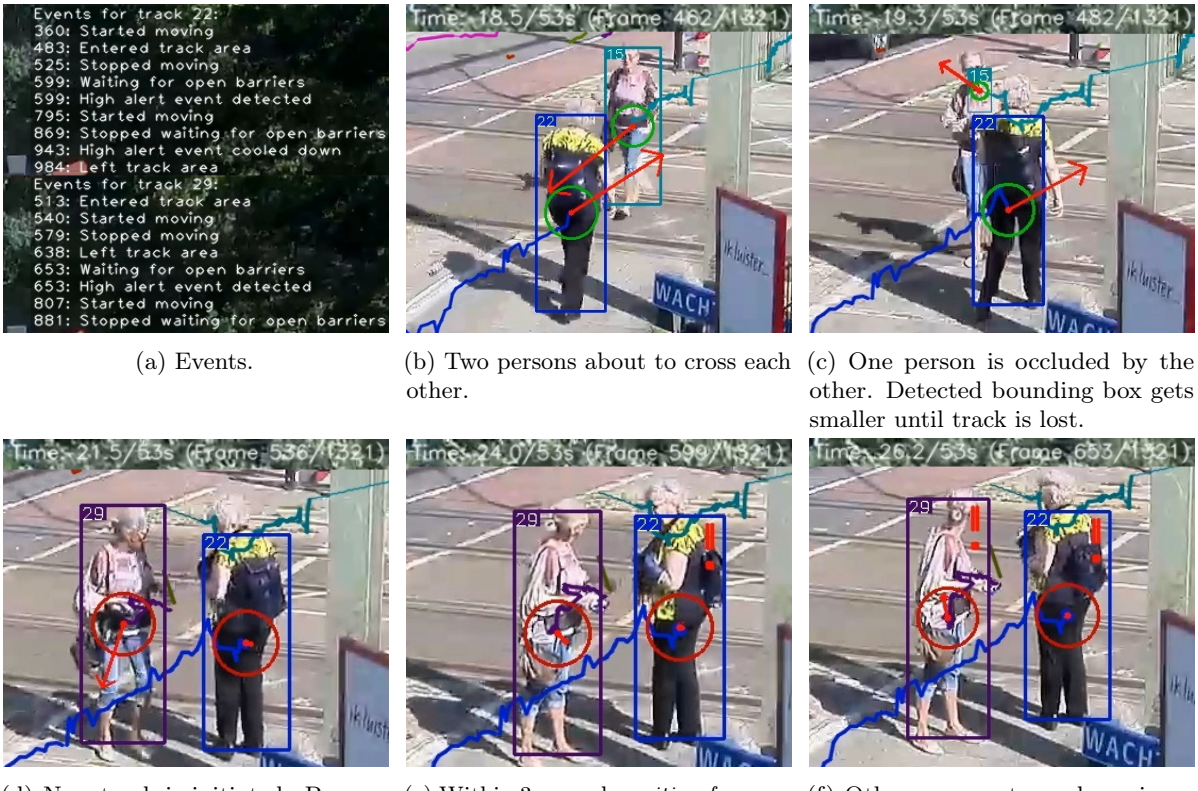
(a) Detected events with frame number.

(b) Person starts waiting while barriers are open.

(c) After three seconds a *waiting for open barriers* event is detected and an alert is triggered.

(d) Person continues on his way, crossing the tracks, and the high alert event cools down.

Figure 31: Evaluation sequence 52.



(a) Events.

(b) Two persons about to cross each other.

(c) One person is occluded by the other. Detected bounding box gets smaller until track is lost.

(d) New track is initiated. Persons stop moving.

(e) Within 3 seconds *waiting for open barriers* event is detected and alert is triggered.

(f) Other person stopped moving a little later. Also triggers an alert.

Figure 32: Evaluation sequence 53.

(a) Police officer walking at brisk pace along tracks. (b) Enters forbidden zone, which is promptly detected.

Figure 33: Evaluation sequence 54.

(a) Detected events with frame number. (b) Person appears to stop moving. (c) Stop event is detected (d) 3 seconds after the person was detected to have stopped moving an alert was triggered because the barriers were open.

Figure 34: Evaluation sequence 55.

(a) Person turned around while barriers were open. Alert triggered. (b) Bounding box size is very jittery. (c) Despite the last turn being detected too early due to the jittery detections, the first turn was correctly detected and the alert was triggered. (d) Events.

Figure 35: Evaluation sequence 56.

References

[1] Cornelis A.J. Van Houwelingen, Ad J.F.M. Kerkhof, and Domien G.M. Beersma. "Train suicides in the Netherlands". In: *Journal of Affective Disorders* 127.1-3 (2010), pp. 281–286.

[2] *ProRail Annual Report 2019*. `http://www.jaarverslagprorail.nl/verslag/spoorprestaties/veiligheid/veilig-leven`.

[3] Athanasios Voulodimos et al. "Deep Learning for Computer Vision: A Brief Review". In: *Computational Intelligence and Neuroscience* 2018 (2018).

[4] C.A.J. van Houwelingen. "Studies into train suicide: The contribution of psychopathology, railway parameters and environmental factors". English. PhD thesis. Vrije Universiteit Amsterdam, 2011.

[5] Brian L. Mishara and Cécile Bardon. "Systematic review of research on railway and urban transit system suicides". In: *Journal of Affective Disorders* 193 (2016), pp. 215–226.

[6] Grigore M. Havârneanu, Jean Marie Burkhardt, and Anne Silla. "Optimizing suicide and trespass prevention on railways: a problem-solving model from the RESTRAIL project". In: *International Journal of Injury Control and Safety Promotion* 24.4 (2017), pp. 469–486.

[7] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2001.

[8] David G. Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the IEEE International Conference on Computer Vision*. 1999.

[9] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*. 2005.

[10] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Machine Learning* (1995).

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Neural Information Processing Systems* 25 (Jan. 2012).

[12] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444.

[13] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* (1986).

[14] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[15] Geoffrey Hinton et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal processing magazine* 29.6 (2012), pp. 82–97.

[16] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[17] Zhong Qiu Zhao et al. "Object Detection with Deep Learning: A Review". In: *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (2019), pp. 3212–3232.

[18] Donggeun Yoo et al. "Attentionnet: Aggregating weak directions for accurate object detection". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

[19] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2014.

[20] J. R.R. Uijlings et al. "Selective search for object recognition". In: *International Journal of Computer Vision* (2013).

[21] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2015).

[22] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2006.

[23] Ross Girshick. "Fast R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2015 Inter. IEEE, Dec. 2015, pp. 1440–1448.

[24] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149.

[25] Jifeng Dai et al. "R-FCN: Object detection via region-based fully convolutional networks". In: *Advances in Neural Information Processing Systems*. 2016.

[26] Tsung Yi Lin et al. "Feature pyramid networks for object detection". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 2017.

[27] Kaiming He et al. "Mask R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision* 2017-Octob (Mar. 2017), pp. 2980–2988.

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

[29] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[30] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[31] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[32] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[33] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[34] Markus Enzweiler and Dariu M. Gavrila. "Monocular pedestrian detection: Survey and experiments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.12 (2009), pp. 2179–2195.

[35] Paul Viola, Michael J. Jones, and Daniel Snow. "Detecting pedestrians using patterns of motion and appearance". In: *International Journal of Computer Vision* (2005).

[36] Christian Wöhler and Joachim K. Anlauf. "An adaptable time-delay neural-network algorithm for image sequence analysis". In: *IEEE Transactions on Neural Networks* (1999).

[37] Antonio Brunetti et al. "Computer vision and deep learning techniques for pedestrian detection and tracking: A survey". In: *Neurocomputing* 300 (2018), pp. 17–33.

[38] Liliang Zhang et al. "Is faster R-CNN doing well for pedestrian detection?" In: *European conference on computer vision*. Springer. 2016, pp. 443–457.

[39] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)". In: *The annals of statistics* 28.2 (2000), pp. 337–407.

[40] Wei Liu et al. "Learning efficient single-stage pedestrian detectors by asymptotic localization fitting". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11218 LNCS (2018), pp. 643–659.

[41] Shanshan Zhang, Jian Yang, and Bernt Schiele. "Occluded Pedestrian Detection Through Guided Attention in CNNs". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018, pp. 6995–7003.

[42] Stephen Siena and B. V.K.Vijaya Kumar. "Detecting occlusion from color information to improve visual tracking". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. Vol. 2016-May. IEEE, Mar. 2016, pp. 1110–1114.

[43] Junhyug Noh et al. "Improving occlusion and hard negative handling for single-stage pedestrian detectors". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 966–974.

[44] Chunluan Zhou and Junsong Yuan. "Bi-box Regression for Pedestrian Detection and Occlusion Estimation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11205 LNCS (2018), pp. 138–154.

[45] Leonid Pishchulin et al. "DeepCut: Joint subset partition and labeling for multi person pose estimation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016.

[46] Eldar Insafutdinov et al. "Deepercut: A deeper, stronger, and faster multi-person pose estimation model". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9910 LNCS (2016), pp. 34–50.

[47] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. "Densepose: Dense human pose estimation in the wild". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7297–7306.

[48] Jianbing Shen et al. "Fast Online Tracking With Detection Refinement". In: *IEEE Transactions on Intelligent Transportation Systems* 19 (Jan. 2018), pp. 162–173.

[49] Alex Bewley et al. "Simple online and realtime tracking". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 3464–3468.

[50] Caglayan Dicle, Octavia I. Camps, and Mario Sznaier. "The Way They Move: Tracking Multiple Targets with Similar Appearance". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2013.

[51] Seyed Hamid Rezatofighi et al. "Joint Probabilistic Data Association Revisited". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[52] Chanho Kim et al. "Multiple Hypothesis Tracking Revisited". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[53] Li Zhang, Yuan Li, and Ramakant Nevatia. "Global data association for multi-object tracking using network flows". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–8.

[54] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. "Globally-optimal greedy algorithms for tracking a variable number of objects". In: *CVPR 2011*. IEEE. 2011, pp. 1201–1208.

[55] Jerome Berclaz et al. "Multiple object tracking using k-shortest paths optimization". In: *IEEE transactions on pattern analysis and machine intelligence* 33.9 (2011), pp. 1806–1819.

[56] Bo Yang and Ram Nevatia. "An online learned CRF model for multi-target tracking". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 2034–2041.

[57] Bo Yang and Ram Nevatia. "Multi-target tracking by online learning of non-linear motion patterns and robust appearance models". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1918–1925.

[58] Anton Andriyenko, Konrad Schindler, and Stefan Roth. "Discrete-continuous optimization for multi-target tracking". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1926–1933.

[59] Anton Milan, Konrad Schindler, and Stefan Roth. "Detection-and trajectory-level exclusion in multiple object tracking". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 3682–3689.

[60] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45.

[61] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[62] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple online and realtime tracking with a deep association metric". In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.

[63] Prasanta Chandra Mahalanobis. "On the generalized distance in statistics". In: National Institute of Science of India. 1936.

[64] Nicolai Wojke and Alex Bewley. "Deep cosine metric learning for person re-identification". In: *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2018, pp. 748–756.

[65] Guanghan Ning et al. "Spatially supervised recurrent convolutional neural networks for visual object tracking". In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2017, pp. 1–4.

[66] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[67] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. "Tracking without bells and whistles". In: *Proceedings of the IEEE international conference on computer vision*. 2019, pp. 941–951.

[68] Zhongdao Wang et al. "Towards real-time multi-object tracking". In: *arXiv preprint arXiv:1909.12605* (2019).

[69] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. "A survey of advances in vision-based human motion capture and analysis". In: *Computer vision and image understanding* 104.2-3 (2006), pp. 90–126.

[70] Ronald Poppe. "A survey on vision-based human action recognition". In: *Image and vision computing* 28.6 (2010), pp. 976–990.

[71] Aaron F. Bobick and James W. Davis. "The recognition of human movement using temporal templates". In: *IEEE Transactions on pattern analysis and machine intelligence* 23.3 (2001), pp. 257–267.

[72] Moshe Blank et al. "Actions as space-time shapes". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. IEEE. 2005, pp. 1395–1402.

[73] Alper Yilmaz and Mubarak Shah. "Actions sketch: A novel action representation". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 984–989.

[74] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. "Going deeper into action recognition: A survey". In: *Image and vision computing* 60 (2017), pp. 4–21.

[75] Ivan Laptev. "On space-time interest points". In: *International Journal of Computer Vision*. 2005.

[76] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. "An efficient dense and scale-invariant spatio-temporal interest point detector". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2008.

[77] Piotr Dollár et al. "Behavior recognition via sparse spatio-temporal features". In: *Proceedings - 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, VS-PETS*. 2005.

[78] Jingen Liu, Jiebo Luo, and Mubarak Shah. "Recognizing realistic actions from videos in the Wild". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*. 2009.

[79] Alexander Kläser, Marcin Marszałek, and Cordelia Schmid. "A spatio-temporal descriptor based on 3D-gradients". In: *BMVC 2008 - Proceedings of the British Machine Vision Conference 2008*. 2008.

[80] Ivan Laptev et al. "Learning realistic human actions from movies". In: *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 2008.

[81] Navneet Dalal, Bill Triggs, and Cordelia Schmid. "Human Detection Using Oriented Histograms of Flow and Appearance". In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 428–441.

[82] Vadim Kantorov and Ivan Laptev. "Efficient feature extraction, encoding, and classification for action recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2014.

[83] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2002).

[84] Guoying Zhao and Matti Pietikäinen. "Dynamic texture recognition using local binary patterns with an application to facial expressions". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007).

[85] Vili Kellokumpu, Guoying Zhao, and Matti Pietikäinen. "Human activity recognition using a dynamic texture based method". In: *BMVC 2008 - Proceedings of the British Machine Vision Conference 2008*. 2008.

[86] Ehsan Norouznezhad et al. "Directional space-time oriented gradients for 3D visual pattern analysis". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2012.

[87] Andres Sanin et al. "Spatio-temporal covariance descriptors for action and gesture recognition". In: *Proceedings of IEEE Workshop on Applications of Computer Vision*. 2013.

[88] João Carreira et al. "Free-form region description with second-order pooling". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2015).

[89] Xiaojiang Peng et al. "Action recognition with stacked fisher vectors". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2014.

[90] Hervé Jégou et al. "Aggregating local descriptors into a compact image representation". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010.

[91] Relja Arandjelovic and Andrew Zisserman. "All about VLAD". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2013.

[92] Yan Zhu et al. "Sparse coding on local spatial-temporal volumes for human action recognition". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2011.

[93] Tanaya Guha and Rabab K. Ward. "Learning sparse representations for human action recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2012).

[94] Yingwei Li et al. "VLAD3: Encoding Dynamics of Deep Features for Action Recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016.

[95] Shuiwang Ji et al. "3D Convolutional neural networks for human action recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013).

[96] Joe Yue Hei Ng et al. "Beyond short snippets: Deep networks for video classification". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2015.

[97] Andrej Karpathy et al. "Large-scale video classification with convolutional neural networks". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2014.

[98] Du Tran et al. "Learning spatiotemporal features with 3D convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

[99] Gul Varol, Ivan Laptev, and Cordelia Schmid. "Long-Term Temporal Convolutions for Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

[100] Lin Sun et al. "Human action recognition using factorized spatio-temporal convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

[101] Melvyn A. Goodale and A.David Milner. "Separate visual pathways for perception and action". In: *Trends in Neurosciences* 15.1 (1992), pp. 20–25.

[102] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos". In: *Advances in Neural Information Processing Systems*. 2014.

[103] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. "Convolutional Two-Stream Network Fusion for Video Action Recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016.

[104] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning*. 2008.

[105] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[106] Xing Yan et al. "Modeling video dynamics with deep dynencoder". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2014.

[107] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. "Unsupervised learning of video representations using LSTMs". In: *32nd International Conference on Machine Learning, ICML 2015*. 2015.

[108] Michael Mathieu, Camille Couprie, and Yann LeCun. "Deep multi-scale video prediction beyond mean square error". In: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*. 2016.

[109] Ross Goroshin et al. "Unsupervised learning of spatiotemporally coherent metrics". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

[110] Xiaolong Wang and Abhinav Gupta. "Unsupervised learning of visual representations using videos". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

[111] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. "Unsupervised Learning using Sequential Verification for Action Recognition". In: *ArXiv* (2016).

[112] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. "Actions ~ Transformations". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016.

[113] Basura Fernando and Stephen Gould. "Learning end-to-end video classification with rank-pooling". In: *33rd International Conference on Machine Learning, ICML 2016*. 2016.

[114] B Ravi Kiran, Dilip Mathew Thomas, and Ranjith Parakkal. "An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos". In: *Journal of Imaging* 4.2 (2018), p. 36.

[115] Sadegh Mohammadi et al. "Angry crowds: Detecting violent events in videos". In: *European Conference on Computer Vision*. Springer. 2016, pp. 3–18.

[116] Waqas Sultani and Jin Young Choi. "Abnormal traffic detection using intelligent driver model". In: *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, pp. 324–327.

[117] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. "Anomaly detection using one-class neural networks". In: *arXiv preprint arXiv:1802.06360* (2018).

[118] Diederik P Kingma and Max Welling. "Stochastic gradient VB and the variational auto-encoder". In: *Second International Conference on Learning Representations, ICLR*. Vol. 19. 2014.

[119] Alireza Makhzani et al. "Adversarial autoencoders". In: *arXiv preprint arXiv:1511.05644* (2015).

[120] Mohammad Sabokrou et al. "Deep-Anomaly: Fully Convolutional Neural Network for Fast Anomaly Detection in Crowded Scenes". In: *Computer Vision and Image Understanding* 172 (Sept. 2016), pp. 88–97.

[121] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[122] Raghavendra Chalapathy and Sanjay Chawla. "Deep learning for anomaly detection: A survey". In: *arXiv preprint arXiv:1901.03407* (2019).

[123] Waqas Sultani, Chen Chen, and Mubarak Shah. "Real-World Anomaly Detection in Surveillance Videos". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018, pp. 6479–6488.

[124] Tsung Yi Lin et al. "Microsoft COCO: Common objects in context". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8693 LNCS.PART 5 (2014), pp. 740–755.

[125] Richard C. Gerum et al. "CameraTransform: A Python package for perspective corrections and image mapping". In: *SoftwareX* 10 (July 2019), p. 100333.

[126] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[127] Joseph Redmon. *Darknet: Open Source Neural Networks in C*. `http://pjreddie.com/darknet/`. 2013–2016.

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

[129] VideoLan. *VLC media player*. 2006.

[130] Boris Sekachev, Andrey Zhavoronkov, and Nikita Manovich. *Computer Vision Annotation Tool: A Universal Approach to Data Annotation*. `http://software.intel.com/content/www/us/en/develop/articles/computer-vision-annotation-tool-a-universal-approach-to-data-annotation.html`. 2019.

[131] Fei Yin, Dimitrios Makris, and Sergio Velastin. "Performance Evaluation of Object Tracking Algorithms". In: *Performance Evaluation*. Vol. 22. 2007, pp. 1–8.

[132] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[133] Greg Welch and Gary Bishop. "An Introduction to the Kalman Filter". In: *Proc. Siggraph Course* 8 (Jan. 2006).

[134] Christian Micheloni, Bernhard Rinner, and Gian Luca Foresti. "Video analysis in pan-tilt-zoom camera networks". In: *IEEE Signal Processing Magazine* 27.5 (2010), pp. 78–90.

[135] Giuseppe Lisanti et al. "Continuous localization and mapping of a pan–tilt–zoom camera for wide area tracking". In: *Machine Vision and Applications* 27.7 (2016), pp. 1071–1085.

[136] Yong Shean Chong and Yong Haur Tay. "Abnormal event detection in videos using spatiotemporal autoencoder". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10262 LNCS (Jan. 2017), pp. 189–196.