Multiple object tracking in complex urban settings

Mid-term report

Authors:
Davide Carrara [*]
Lupo Marsigli [†]
Francesco Romeo [‡]
Valentina Sgarbossa [§]

Examiner:
Alexey Heintz

Supervisors:
Yury Tarakanov
Gustav Tellwe

Project Course in Mathematical and Statistical Modeling
Chalmers University of Technology

January 24, 2022

[*]davide1.carrara@mail.polimi.it
[†]lupo.marsigli@mail.polimi.it
[‡]francesco5.romeo@mail.polimi.it
[§]valentina.sgarbossa@mail.polimi.it

Abstract

Object tracking is the process of detecting and following moving entities in subsequent observations from multiple sensors.

Accurate tracking in complex urban scenarios is crucial for safety applications such as accident mitigation, predictive traffic control and design of safer infrastructure.

Despite a well-established theoretical framework for object tracking, various challenges need to be addressed in a real context, including sensor noise, occlusions and asynchronous measures from multiple sensors.

In this report, a full algorithm to track multiple objects is developed, with focus on accuracy of trajectory estimation.

This part will also include a brief comment on results

Contents

List of Figures

List of Tables

1 Background

Urban settings are traditionally very complicated environments, with high potential for artificial intelligence applications to improve safety and traffic.

Viscando uses proprietary stereovision sensors to extract visual information in critical urban settings, and algorithms to produce insights on mobility risks, communicate real-time safety information and study traffic scenarios and behavior models.

One key feature of these vision tools is the ability to correctly identify and locate objects, and track their movement in consecutive measurements or partially overlapping views. Indeed, prediction of trajectories might be a crucial aid for smart collision warning, or enhanced visibility, just to cite a few.

The aim of the present work is to provide an algorithm to unify detections into tracks, with a focus on accuracy and smoothness of the output, as well as providing a flexible implementation that may easily be modified and re-used. For this reason, we provide three classes, corresponding to the essential concepts that are represented in the algorithm, namely *object, track* and *tracker.* The latter provides the machinery to receive data, update existing tracks, generate new ones and predict position in the next time frame.

1.1 Data

Provided data is a collection of information from 4 stereovision sensors, with detections corresponding to approximately 750 frames or one minute of activity, of a roundabout in Kölliken, Switzerland. Data has been previously processed to provide points in 3D as centers of detected objects, and bounding boxes containing those objects.

Data is organized as follows:

```
root(sensor specific)
├─ Timestamp
├─ TMatrixWorldToCam
├─ TMatrixCamToWorld
├─ ProjectionMatrix
└─ Sequence
   ├─ 0
   │  ├─ Detections
   │  ├─ Image
   │  ├─ Points
   │  ├─ Security
   │  └─ Tracks
   ├─ 1
   │  └─ ...
   └─ ...
```

- **TMatrixWorldToCam, TMatrixCamToWorld** are used to pass object coordinates from the reference system of the sensor to that of the world and vice versa
- **ProjectionMatrix** is used to project 3D coordinates of objects in the bi-dimensional reference system that allows visualization as image
- **Sequence** contains data for every observed frame
- **Detections** contain data about detected points and their attributes. They are provided as a list of
  - $[X, Y, Z]$ coordinates of the center
  - semi-length of the bounding box
  - semi-width of the bounding box

  - semi-height of the bounding box
  - angle of the bounding box with respect to the horizontal axis

- **Image** is a grayscale representation of the observed scene
- **Points** are point clouds corresponding to each detection
- **Security** is a measure of the effectiveness of the stereo vision at a pixel, measured as number from 0 to 255
- **Tracks** are current outputs of Viscando's algorithms, and they should not be used in our algorithm, but can rather be a term of comparison

1.2 Challenges

The main challenges we have faced so far can be divided in two levels: the quality of the detections and the collection of data made by the four sensors.

Regarding the first problem, it can happen that an object detected in a certain frame is occluded in the following ones, resulting in uncertainties about the position of the object. As a consequence, the tracker can encounter difficulties in tracking the object, especially if it has just appeared in the roundabout. In addition to this, the sensors may fail in associating only one detection per frame to an object. The problem of multiple detections is very common in the data set, and can cause several tracks to be associated to a single object. It is thus important to understand which detections to retain. The last issue to tackle about the detections was the presence of false detections, that is, detections not corresponding to pedestrians or vehicles, the main characters of our analysis. Luckily this was a more simple topic to address, given the fact that these false detections usually disappear after one frame.

As we mention before, there are also few issues related to the collection of data from the four sensors, such as the absence of some frames, the different precision that a single sensor has, given the distance of the object from the camera, and the asynchronicity of the frames. With the latter we mean that the different sensors should capture frames in the same time instant and within the same time interval, but they often fail resulting in shifted images.

2 Theory

We report the theoretical framework and literature review for the implemented blocks.

2.1 Intersection over Union

One critical advantage of having bounding boxes for each detection is the possibility to incorporate information about the dimension of an object and volume it occupies, rather than only accounting for the estimated position of its center. Starting from bounding boxes, one may define a metric called *Intersection over Union* for boxes b_1 and b_2:

$$IoU(b_1, b_2) = \frac{Volume(b_1) \cap Volume(b_2)}{Volume(b_1) \cup Volume(b_2)} \tag{1}$$

This metric may be used to build a very simple tracker as in [1]. A track is assigned to the object with the highest IoU if this is higher than a predefined threshold σ_{IoU}. If no object gives an higher IoU than the threshold, the track is terminated, and any object that is not assigned will initiate a new track. Moreover, all tracks shorter than a given number of observations are discarded.

Whereas this algorithm is not addressing the problem of obstructions and may not yield the optimal assignment of tracks, it is very useful as underlying tracker for its simplicity.

We implement our version of the algorithm, using the last available box for tracks that have *pending* status. This method may work acceptably in most cases for our urban setting, as in a roundabout most objects will not change their position too abruptly. However, in the most critical cases boxes may vary very quickly, and possibly become disjoint.

Another potential drawback of using a purely position-based tracker is that the position estimate will be very noisy. Furthermore, when trying to incorporate the information of different viewpoints for the same object (cf. Sec. 3.3), our approach of averaging measures of the boxes might cause substantial errors.

These issues reinforce the need for a statistical tool like Kalman Filter that can account for noise and errors and provide predictions even without observed data. Nevertheless, IoU remains a valid instrument to complement the filter, especially as it is generally more precise in the association of tracks to detections and since it can provide aid in the transitory phase in which the filter has not yet converged to the stationary value of covariance of prediction error.

Therefore, the following metric is used to link tracks to detections (using a greedy approach as in [1]):

$$d(track, det) = \omega_1(1 - IoU(box_{track}, box_{det})) + \omega_2 \|x_{\hat{track}} - x_{det}\|_{L^2} \tag{2}$$

where IoU is as in (1) and $x_{\hat{track}}$ is the predicted center position through Kalman Filter at the next time step. Note that the weights ω_1 and ω_2 need to be calibrated to yield an optimal result.

2.2 Kalman Filter

The Kalman Filter is a recursive linear estimator which successively calculates an estimate for a continuous valued state, that evolves over time, on the basis of observations of a variable of interest. The Kalman Filter employs an explicit statistical model of how the parameter of interest $x(t)$ evolves over time and an explicit statistical model of how the observations $y(t)$ that are made are related to this parameter.

The starting point for the Kalman filter algorithm is to define a model for the states to be estimated in the standard state-space form:

$$\begin{cases} x(t+1) = Fx(t) + Gu(t) + v_1(t) \\ y(t) = Hx(t) + Du(t) + v_2(t) \end{cases} \tag{3}$$

where: v_1 and v_2 are white noises, that is, a stationary random process with zero autocorrelation, F is known as state matrix and it depends on the motion model adopted, H is the observation matrix, G and D are input matrices.

In particular, for our analysis we will neglect the effects of external inputs putting the matrices G and D equal to the null matrix, and we will consider as state variables the position, speed, and acceleration of the objects in the three direction of the space.

The assumptions underlying this model are multiple:

- No external inputs, $Gu(t)$ and $Du(t)$ are equal to the zero vector
- The system should be linear and time invariant
- v_1 and v_2 are vectorial white-noises such that:

  $$\mathbb{E}[v_i(t)] = 0, \quad \mathbb{E}[v_i(t)v_i(t)^T] = V_i, \quad \mathbb{E}[v_i(t)v_i(t-\tau)^T] = 0 \quad \forall t,\ \forall \tau \neq 0 \tag{4}$$

  where $i = 1, 2$ and V_i is a positive semidefinite and symmetric matrix
- The following relationship holds between v_1 and v_2:

  $$\mathbb{E}[v_1(t)v_2(t-\tau)^T] = V_{12} = \begin{cases} 0 & \text{if } \tau \neq 0 \\ \text{can be non-zero} & \text{if } \tau = 0 \end{cases} \tag{5}$$

Due to the multiple advantages this filter has, we chose the Kalman filter algorithm as main feature of the tracker we will implement. First of all, the K.F. requires no training data, which is convenient for us since we don't have much data available, neither a ground truth of the state of the vehicles or of the pedestrians. In addition to this, the Kalman Filter is able to smooth trajectories filtering the noise contained in the detections. This algorithm in fact makes, as first step, a prediction of the future state of the model, called $\hat{x}(t+1|t)$ and then, through the given observation $y(t)$, filters the state, obtaining $\hat{x}(t|t)$ which is an estimate of the true (but unknown) state of the object. In particular, the equations that describe the prediction process mentioned above are the following: at iteraton time t

$$\hat{x}(t+1|t) = F\hat{x}(t|t-1) + K(t)e(t) \qquad \text{state equation} \qquad (6)$$

$$\hat{y}(t|t-1) = H\hat{x}(t|t-1) \qquad \text{output equation} \qquad (7)$$

$$e(t) = y(t) - \hat{y}(t|t-1) \qquad \text{output prediction error} \qquad (8)$$

$$K(t) = \left(FP(t)H^{\mathrm{T}} + V_{12}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1} \qquad \text{gain of KF} \qquad (9)$$

where $P(t)$ is the most important matrix of the KF together with the gain of the filter $K(t)$, and his expression is given by the difference Riccati equation:

$$P(t+1) = (FP(t)F^T + V_1) - (FP(t)H^T + V_{12})(HP(t)H^T + V_2)^{-1}(FP(t)H + V_{12})^T \qquad (10)$$

Equations (6) and (10) are dynamic equations and thus need initial conditions:

$$\hat{x}(1|0) = \mathbb{E}[x(1)] = X_0 \qquad P(1) = \mathrm{Var}[x(1)] = P_0 \qquad (11)$$

The initial condition of matrix $P(t)$ is linked to an important property of this matrix:

$$P(t) = \mathrm{Var}[x(t) - \hat{x}(t|t-1)] \qquad (12)$$

that is, $P(t)$ is the covariance matrix of the 1-step prediction error of the state.

Furthermore, the equations that characterise the filtering process, which allows to obtain $\hat{x}(t|t)$, are the following: at iteration time t

$$\hat{x}(t|t) = F\hat{x}(t-1|t-1) + K_0(t)e(t) \qquad (13)$$

$$K_0(t) = \left(P(t)H^{\mathrm{T}}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1} \qquad (14)$$

where $e(t)$ has the same definition as in equation (8). These equations are valid under the restrictive (but very common) assumption $V_{12} = 0$.

2.3 Motion Model

Kalman Filter assumes complete knowledge of the physical laws governing the modeled phenomenon. In our case, one may conjecture different models for different object types on the road: a car will move differently from a bicycle, and both have a behavior that is not like the one of pedestrians. A relatively flexible model is that of Constant Acceleration as reported in [2].

The main advantage of this simple model is that it is linear, and therefore convergence theorems of Kalman Filters apply. However, the approximation does not include varying accelerations, as for instance yaw rate.

The model is as follows:

$$x(t) = \begin{bmatrix} x(t) \\ v_x(t) \\ a_x \\ y(t) \\ v_y(t) \\ a_y \\ z(t) \\ v_z(t) \\ a_z \end{bmatrix} \qquad F = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}(\Delta t)^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

3 Algorithm

3.1 Introduction

The proposed algorithm is based on three main classes, representing the three fundamental physical entities.

The Object class represents the physical object, usually a vehicle, a cyclist or a pedestrian that is observed by the stereo sensors. Every detection by the sensor at first defines an object, with the problem of multiple detections discussed in the following subsection 3.3.

The class Track contains all the information necessary to define the motion dynamics, namely the states and matrices of the Kalman Filter. This class also shares some important attributes such as the state of the Track and the time spent in that state.

The class Tracker contains the methods and the information necessary for the main task of the project, namely:

- Connect existing tracks with new detections
- Merge tracks corresponding to the same physical objects or potentially divide tracks belonging to overlapping objects
- Evaluate and update the state of the tracks
- Provide visualization for the obtained trajectories

3.2 Asynchronicity

The first issue to tackle is the asynchronicity of the four sensors. In particular, the sensors do not start detecting objects at the same time, and the time-step between each frame is slightly different, even if always close to 0.08 s. Moreover, it happens sometimes that a sensor loses a frame, meaning that the detection for an expected instant is not present in our dataset.

There are essentially two possible approaches to handle this problem:

- Interpret each frame separately, creating a flow of detections coming, for each update, from a single sensor.
- Use an internal clock for the tracker, with a fixed update time. In this way every object detected in the time preceding the update is considered to be observed at the same time instant.

Even if the former approach might appear more desirable, the latter has been preferred for multiple reasons. Firstly, this procedure is coherent with systems available in commerce, with a small but fixed update time. Moreover, this approach allows, for a small enough time-step, to reduce the impact of errors in the coordinates detection, due to low precision of the sensors in some areas of the roundabout, as explained in 3.3.2. It has been possible to observe that, for small time differences, the advantages brought by averaging the coordinates of the detections of same objects are greater than the loss of information due to very small movements. Lastly, this approach can be refined into the first one, by taking a very small update time, once a more performing and advanced model is available.

In the specific case, the Tracker implemented at the moment starts with a time clock set at -0.06 s and has an update time of 0.08 s. It has been suggested, during the last meeting with the Supervisor, to reduce this update time to take into account possible fast movements of incoming cars.

3.3 Merge detections from different sensors

Once the time of the internal clock of the tracker is fixed, and the corresponding frames belonging to the different sensors are selected, the problem of multiple detections arises. This consists in the same object being detected multiple times, at slightly different positions, and thus being interpreted by the tracker as two different items. It is possible to find two main causes for this issue:

- Frames collected by different sensors are not completely synchronous and thus the object is correctly represented in two different positions. Still, the time difference is small and negligible in our analysis.
- Even when the sensors are completely simultaneous, they come with a margin of error, which leads to different measurements of the same object, both in terms of coordinates and bounding boxes. Moreover, these margins of error are not constant among all the identified objects, but vary according to the specific sensor and the area observed.

  In order to tackle this problem it has been necessary to develop a system to identify the detections belonging to the same object and, after that, to establish rules to condense the obtained information in a unique instance.

3.3.1 Identification

We introduced a new similarity measure to compare two detections, and thus understand if they belong to the same object. The similarity measure $h(\mathbf{a}, \mathbf{b})$ works by setting a multivariate gaussian distribution centered on $\mathbf{a}$ and computing the probability of obtaining a detection with greater or equal Mahalanobis distance than the one between $\mathbf{a}$ and $\mathbf{b}$.

In particular, we define

$$\mathbf{a} = [\ \mathrm{x},\ \mathrm{y},\ \mathrm{z}\ ]^T$$
$$\bar{\mathbf{a}} = [\ \mathrm{x},\ \mathrm{y},\ \mathrm{z},\ \text{box length},\ \text{box width},\ \text{box height},\ \text{timestamp},\ \text{camera}\ ]^T$$

We define the multivariate gaussian related to $\mathbf{a}$ with quantities

$$\mathbf{mean} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \mathbf{covariance} = \Sigma = \begin{bmatrix} \text{box\_length}^2 & 0 & 0 \\ 0 & \text{box\_width}^2 & 0 \\ 0 & 0 & \text{box\_height}^2 \end{bmatrix} * \exp\left(\frac{|\Delta \mathrm{t}|}{0.05} \right)$$

The covariance matrix is generated by the dimensions of the boxes and is then multiplied by a coefficient related to the time distance between the two detections.

Then Mahalanobis distance is computed as

$$\mathrm{d_m}(\mathbf{a}, \mathbf{b}) = (\mathbf{a} - \mathbf{b})^T\ \Sigma^{-1}\ (\mathbf{a} - \mathbf{b}) \tag{15}$$

and its distribution is known to be $\sim \chi^2(3)$.

We can then finally compute

$$h(\mathbf{a}, \mathbf{b}) = \begin{cases} 0 & \textbf{if}\ \ \bar{a}_{\text{camera}} = \bar{b}_{\text{camera}} \\ 1 - c.d.f.(\mathrm{d_m}(\mathbf{a}, \mathbf{b})) & \textbf{else} \end{cases} \tag{16}$$

This value indicates a level of confidence of detection $\mathbf{b}$ belonging to the same object represented by detection $\mathbf{a}$. This similarity measure is of course not symmetric, since it takes into account the bounding box of only one detection in the computation.

Given a frame, the similarities of the objects are stored in matrix $\mathbf{C}$, such that $\mathbf{C}_{\mathrm{i,j}} = h\,(\,\mathbf{obj_i},\,\mathbf{obj_j}\,)$

The algorithm to associate detections proceeds then in this way

Algorithm 1 Identification Algorithm

```
index[i] ← None  $\forall i \in \mathcal{I}$        ▷ Detections with same index represent the same object
$i^*, j^* \leftarrow \arg\max_{i,j} \mathbf{C}$, $c^* \leftarrow \max_{i,j} \mathbf{C}$
while $c^* \geq 0.5$ do
    if $i^*$ != $j^*$ and $\mathbf{C}[j^*, i^*] \geq 0.1$ then        ▷ C is not symmetrical
        if index[$j^*$] is None then
            if index[$i^*$] is None then index[$i^*$] ← $i^*$
            index[$j^*$] ← index[$i^*$]
    $\mathbf{C}[i^*, j^*] \leftarrow -1$
    $i^*, j^* \leftarrow \arg\max_{i,j} \mathbf{C}$, $c^* \leftarrow \max_{i,j} \mathbf{C}$
for all $i \in \mathcal{I}$ do
    if index[i] is None then index[i] ← i
```

3.3.2 Credibility Estimation

Once the identification process described above is done, we decided to give to each detection of each sensor a level of credibility, based on the position where the detection is made and the camera that did it. The idea behind is that, given a certain sensor, there are some areas of the roundabout where the camera can do a more precise identification of the object and some where, due to the presence of obstacles, the camera may either fail in detecting the target or duplicate the detection for a single object. For all these reasons, we proceeded dividing the coordinates space of each image in different regions, based on how the sensor behaves in detecting object from that area over different frames. We then gave to each a credibility level, that is a number between 0 and 1. Below (Fig. 1) we reported four images taken in different time instants from each of the four sensors respectively.

3.3.3 Object Merging

Once the grouping of different detections into unique objects is available, and each detection has been associated to a credibility level as described above, it is possible to actually merge them. This is performed by taking a weighted average of the coordinates and bounding boxes of the detections belonging to the same object, having as weights the normalized credibility levels.

3.4 Tracker

The *Tracker* class holds *Track* objects grouped by status (*Active, New, Pending, Removed, Inactive*). The tracker implements all methods necessary to update such groupings as time passes and more data is collected from sensors.

Ideally, one track is initiated when a new object first appears in any of the sensors' detections. The status remains *New* until a fixed number of consecutive frames N_{new} are observed for that track, and it then becomes *Active*. If the track is not updated at any step before reaching N_{new} observations, the track becomes *Removed*.
An *Active* track may become *Pending* if it is not linked at some steps, and it may then return *Active* or become *Inactive*, depending on whether it can be linked to some object in the following $N_{pending}$ time steps or not.

In summary, at each time step all *Active, New* and *Pending* will be compared to the list of detections to find matches, while *Inactive* and *Removed* tracks will not be used again.

Tracker also provides methods to iterate the analysis for a number of frames (advance clock, preprocess data as in Sec. 3.3, link tracks and detections) in online mode, namely by only processing data relative to the current time interval.

Tracks and detections are linked through the following logic: a matrix of pairwise distances (track,detection) is computed with the metric (2). Since the initial estimate of the internal state may be critical, in the first N_{new} frames of a track the state is updated manually with the centers provided by detections, as well as velocities and accelerations estimated with finite differences ($N_{new} \geq 3$)

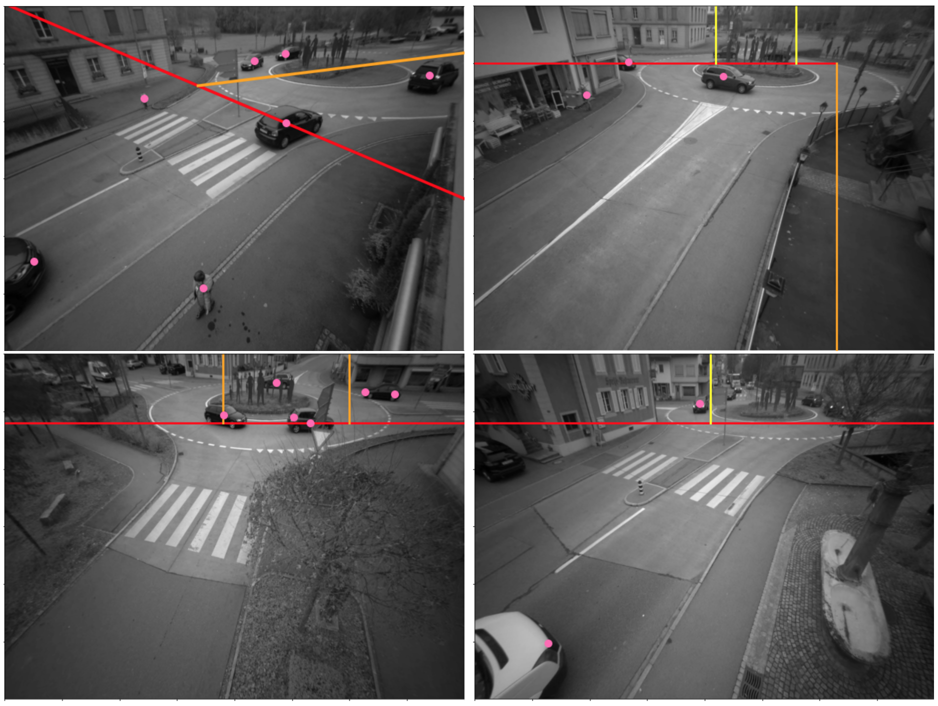

Figure 1: Separation between confidence regions

(in the case of the constant acceleration model). Having obtained the distances of all pairs track-detection, the track with smallest distance is associated to every object. All of these "best" distances are compared to a threshold d_{thr}, such that detections with higher values are automatically set to new objects. If two objects are best matched to the same track, the one with lowest distance is chosen. Finally, all tracks that have been assigned to an object are updated accordingly, while the others are set as *Pending, Removed* or *Inactive.*

4 Results

This section will contain the complete results of the tracker in the final report.

5 Next Steps

The first main future development on our work will focus on establishing some performance indices, in order to fully evaluate the results of our algorithm. Since we do not have available ground truths about trajectories, evaluating the distance between the desired output and the real one is a difficult task. After discussing with our Supervisor, we were suggested to focus on cars, and implement a more visual approach to check for realistic trajectories of vehicles.

It will be fundamental to work on tuning the parameters and thresholds, as the time a track spends in each status or the time-step of the internal clock. However, particular attention will be given to the parameters of the Kalman Filter, which are of fundamental importance in the creation and definition of trajectories.

Some issues with accuracy are probably connected with the the motion model, which considers a constant acceleration when creating and predicting tracks. A future possible development would include trying out more complex motion models, which can better represent specifics of different vehicles.

A possible simple improvement of the algorithm could include establishing confidence levels from how likely a specific detection is representing a new object or one that has already been tracked. This could be done exploiting the coordinates of the detection itself: a new observation in the center of the roundabout will be very unlikely to represent a new object, while one close to the borders has a higher probability.

On a general level, the most significant improvement of our work could be made by applying a Bayesian approach, through multiple hypotheses tracking. In this case, we would not select only the most favorable track at every frame, but keep in memory all possible combinations of possible tracks that meet some credibility criteria.

6 References

[1] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2017.

[2] Robin Schubert, Eric Richter, and Gerd Wanielik. Comparison and evaluation of advanced motion models for vehicle tracking. In *2008 11th International Conference on Information Fusion*, pages 1–6, 2008.