Multiple object tracking in complex urban settings

Project 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 30, 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 filtering noise, as well as producing correct tracking.

The results show that a simple tracker is able to produce similar tracks to the offline benchmark in terms of number and length, although it still needs improvement to achieve comparable levels of smoothness.

The observation of individual cases inspires some proposals for improvements and future work, such as merging tracks referring to the same object, using a multiple hypothesis tracking approach and experimenting with different tracking techniques, for example based on reinforcement learning.

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 detailed traffic movement 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 situation awareness for connected smart vehicles, 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.

In the following, we describe the dataset, explain the theoretical background for the implemented algorithms, introduce some variations motivated by the empirical data and analyse the resulting tracks. Finally, we discuss strengths and weaknesses of the proposed approach and introduce key areas of improvement for future work.

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.

  Normally, Viscando sensors do not collect video data - instead, image frames are removed directly after object detection step. This ensures GDPR compliance of Viscando traffic measurement solutions.

  In the particular case, the data was collected in Switzerland where video collection for development purposes was not prohibited at the time of collection. Collected video (consisting of image frames) is solely used for research and improvement of object detection and tracking algorithms.

- **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 [2]. 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. 4.3), 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.

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:

$$\begin{cases} x(t+1) = Fx(t) + Gu(t) + v_1(t) \\ y(t) = Hx(t) + Du(t) + v_2(t) \end{cases} \tag{2}$$

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:

$$\mathbb{E}[v_i(t)] = 0, \quad \mathbb{E}[v_i(t)v_i(t)^T] = V_i, \quad \mathbb{E}[v_i(t)v_i(t-\tau)^T] = 0 \quad \forall t,\ \forall \tau \neq 0 \tag{3}$$

  where $i = 1, 2$ and V_i is a positive semidefinite and symmetric matrix

- The following relationship holds between v_1 and v_2:

$$\mathbb{E}[v_1(t)v_2(t-\tau)^T] = V_{12} = \begin{cases} 0 & \text{if } \tau \neq 0 \\ \text{can be non-zero} & \text{if } \tau = 0 \end{cases} \tag{4}$$

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.Indeed this algorithm 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

$$\hat{x}(t+1|t) = F\hat{x}(t|t-1) + K(t)e(t) \qquad \text{state equation} \tag{5}$$
$$\hat{y}(t|t-1) = H\hat{x}(t|t-1) \qquad \text{output equation} \tag{6}$$
$$e(t) = y(t) - \hat{y}(t|t-1) \qquad \text{output prediction error} \tag{7}$$
$$K(t) = \left(FP(t)H^{\mathrm{T}} + V_{12}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1} \qquad \text{gain of KF} \tag{8}$$

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:

$$P(t+1) = (FP(t)F^T + V_1) - (FP(t)H^T + V_{12})(HP(t)H^T + V_2)^{-1}(FP(t)H + V_{12})^T \tag{9}$$

Equations (5) and (9) are dynamic equations and thus need initial conditions:

$$\hat{x}(1|0) = \mathbb{E}[x(1)] = X_0 \qquad P(1) = \mathrm{Var}[x(1)] = P_0 \tag{10}$$

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

$$P(t) = \mathrm{Var}[x(t) - \hat{x}(t|t-1)] \tag{11}$$

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

$$\hat{x}(t|t) = F\hat{x}(t-1|t-1) + K_0(t)e(t) \tag{12}$$
$$K_0(t) = \left(P(t)H^{\mathrm{T}}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1} \tag{13}$$

where $e(t)$ has the same definition as in equation (7). 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 [5].

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.

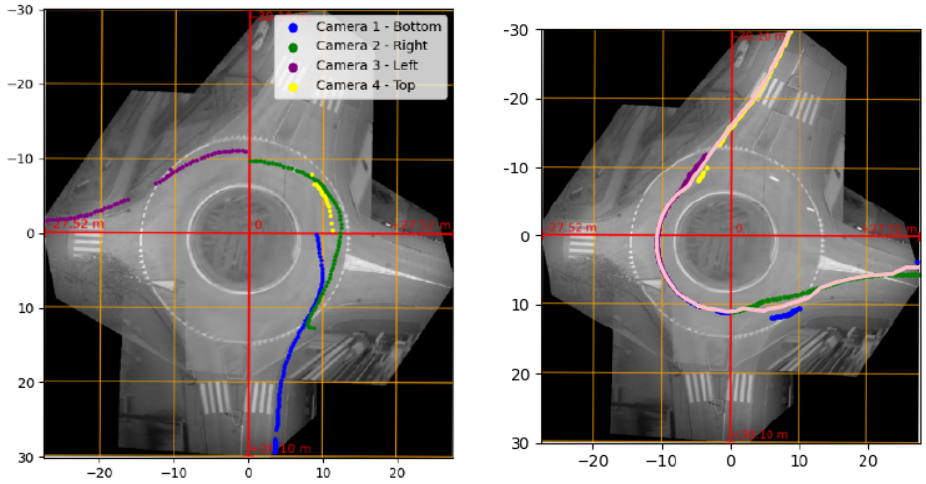


Figure 1: An image of the roundabout from above, before and after applying the proposed transformation

The model is as follows:

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

As we can see, we decided to discard the vertical speed and acceleration, putting them equal to zero, since they are mostly driven by noise in the data. This is also consistent with the motion of the vehicles in the reality.

3 Preprocessing

3.1 Coordinate Transformation

Small discrepancies between the coordinate systems of the four sensors had been observed during the analysis of detections. In particular, estimation of position made by sensor 142 (the left one in Figure 1) proved to be slightly flawed. Another group working on the same dataset proposed thus the following transformation of the coordinates of detections of sensor 142:

- All the detections have to be shifted of + 3 meters in the x-axis direction
- All the detections having radial distance < 10 from the origin have to be radially shifted to distance 10

This transformation resulted in a slight improvement in the perceived precision of our tracking algorithm, and we decided to maintain it.

3.2 Credibility Estimation

A second adjustment that we made consists of giving each detection of each sensor a level of credibility, based on the position where the detection is taken. The idea behind is that, given a certain

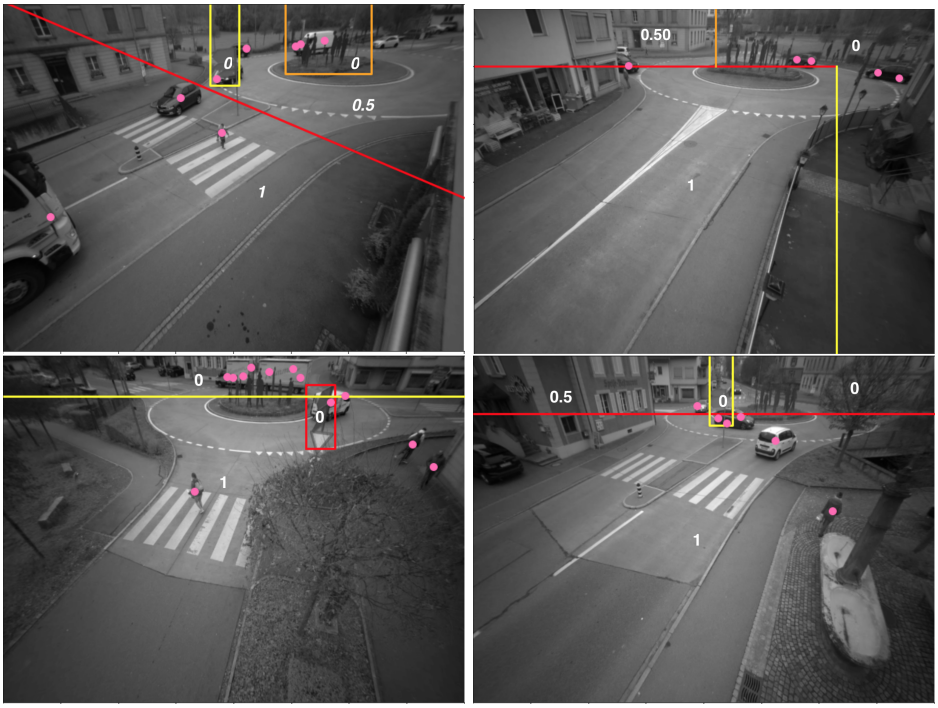


Figure 2: Separation between confidence regions

sensor, there are some areas of the roundabout where this sensor can do a more precise identification of the object and some where, due to the presence of obstacles, the sensor 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 objects from that area over different frames. We then gave to each region a credibility level, that is a number between 0 and 1. Below (Fig. 2) we reported the division in areas we made for each sensor.

A certain weight corresponds to each area in the pictures. At the beginning of our analysis, we decided to give the smallest weight to the most problematic areas, that is, the ones characterized by the presence of permanent obstacles. After further considerations, we realized that the detections related to those areas are often unprecise, resulting in inaccuracies that worsen the outcome of our algorithm. For this reason, we proceed discarding those detections, i.e. giving them null weight. This approach can be justified by the fact that it is possible to find a better detection from one of the other sensors, thus it is better to not consider the noisy detection from one sensor and rely most entirely on the other ones.

The images represent, going from top left to bottom right, sensor 109, 130, 142, and 143. In these images, the problematic areas we mentioned before are:

- Sensor 109: yellow and orange rectangle
- Sensor 130: upper-right area delimited by orange, red and yellow line
- Sensor 142: red rectangle and area above the yellow line
- Sensor 143: yellow rectangle and area above the red line, on the right of the rectangle

In all of these areas, as we can see, the presence of road signs or of the statues in the middle of the roundabout cause unprecise or multiple detections. The numbers in the pictures represent the weights that we used in our algorithm.

Figure 3: Road and Sidewalk division for sensor 109 and 130

3.3 Road and Sidewalk Distinction

The third main adjustment consists in defining a rough division between the sidewalk and the road, in order to separate most of the paths of vehicles and pedestrian. In Figure 3 the division is reported for two sensors. In general, the possibility of identifying the belonging of every object to either the walking or vehicle lane will prove helpful in avoiding the creation of strange mixed tracks, that could happen in the case of a pedestrian exiting the visual field of a sensor in the same moment that a car enters into it.

Even if this approach has been applied only in a naive version, in which only the shapes of road and sidewalk are identified, it is possible and probably useful to implement a more refined version: a sensible approach would include differentiating various lanes and crosswalks, where the behaviour of vehicles may differ from the usual.

4 Algorithm

4.1 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. At first, every detection defines an object, with the problem of multiple detections discussed in the following subsection 4.3.

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

4.2 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 3.2. 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 that in case fast vehicles are present, the update time might need to be smaller to avoid introducing larger position errors.

4.3 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.

4.3.1 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

$$\mathbf{a} = [\ \mathrm{x},\ \mathrm{y},\ \mathrm{z}\ ]^T$$

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

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

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

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

$$d_m(\mathbf{a}, \mathbf{b}) = (\mathbf{a} - \mathbf{b})^T \, \Sigma^{-1} \, (\mathbf{a} - \mathbf{b}) \tag{14}$$

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

We can then finally compute

$$h(\mathbf{a}, \mathbf{b}) = \begin{cases} 0 & \textbf{if } \overline{a}_{\text{camera}} = \overline{b}_{\text{camera}} \\ 1 - c.d.f.(d_m(\mathbf{a}, \mathbf{b})) & \textbf{else} \end{cases} \tag{15}$$

This value indicates a level of confidence of detection **b** belonging to the same object represented by detection **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 $\mathbf{C}_{i,j} = h(\mathbf{obj_i}, \mathbf{obj_j})$

The algorithm to associate detections proceeds then in this way

Algorithm 1 Identification Algorithm

```
index[i] ← None  ∀i ∈ 𝓘                ▷ Detections with same index represent the same object
i*, j* ← arg max_{i,j} C, c* ← max_{i,j} C
while c* ≥ 0.5 do
    if i* != j*  and C[j*, i*] ≥ 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*]
    C[i*, j*] ← −1
    i*, j* ← arg max_{i,j} C, c* ← max_{i,j} C
for all i ∈ 𝓘 do
    if index[i] is None then index[i] ← i
```

4.3.2 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 in Sec.(3.2), 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.

4.4 Computing Distances

Object merging is important because it allows us to refer to a unique detection that contains multiple ones, that is, the ones that have been merged. This unique detection represents an object seen by four different cameras, so it can be used to compute its distance from tracks (taken from a subset of the existing ones) in order to decide to which link it. Particularly, given a set of tracks and detections, the following metric is used to link them:

$$d(track, det) = \omega_1(1 - IoU(box_{track}, box_{det})) + \omega_2 \|\hat{x_{track}} - x_{det}\|_{L^2} \tag{16}$$

where IoU is as in (1) and $\hat{x_{track}}$ is the last position of the track (either a detection, or the status filtered by Kalman Filter as we will see later). One alternative approach would be to project through Kalman Filter (whenever available) and use that position for association instead.

Note that the weights ω_1 and ω_2 need to be calibrated to yield an optimal result.

Having obtained the distances of all pairs track-detection, we can give them as input of the *Visiting Algorithm*.

4.5 Visiting Algorithm

The *Visiting Algorithm* is widely used for matching problems, and its popularity is due to the fact that it can guarantee the best match for at least one of the two categories to be matched. In our case, we used the distances between tracks and detections to establish which assignments to do. Indeed, the matrix of distances computed using the metric (16) is given as input to this algorithm, whose steps are reported below:

1. Assign each detection to the closest track. This corresponds to compute the minimum by column of the distance matrix (assuming that the detections represent the columns).
2. For each track, select the best detection among the ones associated in the step before, that is, the one with the smallest distance, as long as this distance is under a certain *threshold*. The other detections are redirected to the next closest track.
3. If every track as at maximum one association, then the algorithm ends. Otherwise go back to step 2.

The last step inquires that every track can have either one detection linked or no one at all. This happens because we decided to regulate the matching problem using a fixed *threshold*, as mentioned in the second step above. This *threshold* is equal to the minimum entry in the given matrix plus a fixed constant, that we posed equal to 5:

$$d_{thr} = \min_{i,j}(D) + 5 \tag{17}$$

where d_{thr} is the *threshold* and D is the distance matrix given as input. We used this threshold because we preferred to not assign a track than to assign it to a detection which is unlikely to represent the next step of the track.

To conclude, it is worth to mention that this algorithm, carried out as above, guarantees the best match for the detections, and it gives as output a list of associations that can be now used to update the status of the tracks.

4.6 Updating of the tracks

The output of the *visiting algorithm* is important to establish the status of a track. The categories in which we divided the tracks are:

- **New Tracks**: These are the tracks created very recently, of which we are unsure about their entities: they can be both existing objects or false detections

- **Active Tracks**: Contrary to the *New* ones, these tracks are very likely to be objects in the roundabout since they have been associated to a detection for several consecutive frames
- **Pending Tracks**: These are tracks that have not been linked with a detection for few frames, so we consider them as *Pending* for some iterations in order to understand if they are occluded objects or objects that left the roundabout
- **Inactive Tracks**: These should represent objects that have been in the roundabout before leaving it
- **Removed Tracks**: Either false detections or multiple detections for one single vehicle or pedestrian

At the end of the *visting algorithm*, there can exist both tracks and detections that are not matched. Each detection not assigned is the starting point for a *New track*. Regarding tracks, their status and state is updated as follows:

- New Track:
  1. Detection Assigned: If the track has been *New* for more than a fixed *new time threshold*, then it is set to active, otherwise it remains *New*. In both cases, the new detection is used as next point of the track, and speed and acceleration of the track are computed using finite differences considering all the previously associated detections.
  2. Detection Not Assigned: If the track has been *New* for less than *remove time* frames, it becomes *Removed*, otherwise the status is set to *Pending*.
- Active Track:
  1. Detection Assigned: If the track has been *Active* for less than a fixed *active time threshold*, then the detection is used to establish new position, speed and acceleration of the track; particularly, the last two are computed by finite differences. Otherwise the filter comes into play and the filtered state is used to update position, speed and acceleration of the track.
  2. Detection Not Assigned: The status is set to *Pending*.
- Pending Track:
  1. Detection Assigned: The status is set to *Active* and the new position, speed and acceleration are computed in the same way as they were before the tracks became *Pending* (either *New* or *Active*).
  2. Detection Not Assigned: If the track has been *Pending* for more frames than *max pending time*, the track becomes *Inactive*. Otherwise the track remains *Pending* and its next position, speed and acceleration are the ones predicted by the Kalman Filter

Regarding the *New tracks*, since the initial estimate of the internal state may be critical, in the first N_{new} frames the state is updated without the filter, nonetheless we make the filter proceed in order to help it reach convergence when the track has been *Active* for more than *active time thresholds* iterations.

It is important to mention that the tracks *Inactive* and *Removed* are not considered in the computation of the distances.

Moreover, accelerations estimates were found to be prone to large errors due to noise in the observations, which is amplified in the differentiation. Therefore we add a constraint $|a_i| < g, i = x, y, z, g = 9.8m/s^2$ to limit non-physical estimates that might cause the filter to take longer to converge.

4.7 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. Particularly, *Tracker* provides methods to iterate the analysis for a number of frames in online mode, namely by only processing data relative to the current time interval.

We will now describe a single iteration of this *Tracker*.

At the beginning, the *Tracker* determines which frames are synchronous, as described in Sec(4.2), so that the *Tracker* knows which detections it should attempt to merge and from which frames (Sec.4.3). Once these steps are performed, the *Tracker* can proceed computing the distances between detections and *New, Active* and *Pending* tracks. As we can see the *Inactive* and *Removed* tracks are not considered for this phase since they probably do not represent objects in the roundabout.

The Distance matrix is computed following the procedure of Sec.(4.4), and it is given as input to the *Visiting algorithm*, presented in Section (4.5). The last step of the algorithm consist in updating both the status and the state of the track, that is, position, speed and acceleration. This critical step is carried out as in Sec.(4.6).

4.8 Model parameters and tuning

The algorithm accounts for the possibility of different scenarios by allowing for the following parameters to be tuned:

- *dt*: internal clock of the tracker. It was chosen to keep it close to the sensors' acquisition rate 0.08. However, the parameter may be adjusted to take updates with higher or lower frequency, thereby determining the frequency at which updates of the filter are made, and tracks and detections are associated. In principle, all measurements from asynchronous cameras could update the state independently. However, in the studied use-case, where different sensors contribute with unevenly accurate measurements, it was more robust to join the measurements as one at merging stage.

- *new time threshold*: it sets how long the tracks remain *New* since their first detection. We chose to set this to 7 frames.

- *active time threshold*: it sets how long the warm-up for the Kalman Filter lasts (state is projected but not yet considered as estimate of the track position). We chose to set this to 5 frames.

- *speed acc window*: length of the moving window for initial estimate of acceleration and speed. It was set to 5.

- *max dist threshold*: this distance sets a soft maximum to the distance that a track and detection need to have in order to be considered for association. This amount is further added to the minimum distance track-detection for the current iteration, representing the actual threshold until which distances are considered. The rationale for this choice is to adjust how conservative association is also in relation to the observed detections and their noise.

- P_0: the state error initial covariance matrix. This matrix should be anything different from 0, otherwise no learning will happen. It was set to $0.1I_n$, where I_n is the identity matrix with $n = 9$.

- Q: process noise covariance matrix. This was eventually set to $50I_n, n = 9$, after experimenting with smaller values up to $0.005I_n$ and concluding the smoothing is not optimal in those cases.

- R: measurement noise covariance matrix. This was eventually set to $diag(80, 150, 200)$, after experimenting with smaller values up to $0.01diag(0.8, 1.5, 2)$ and concluding the smoothing is not optimal in those cases.

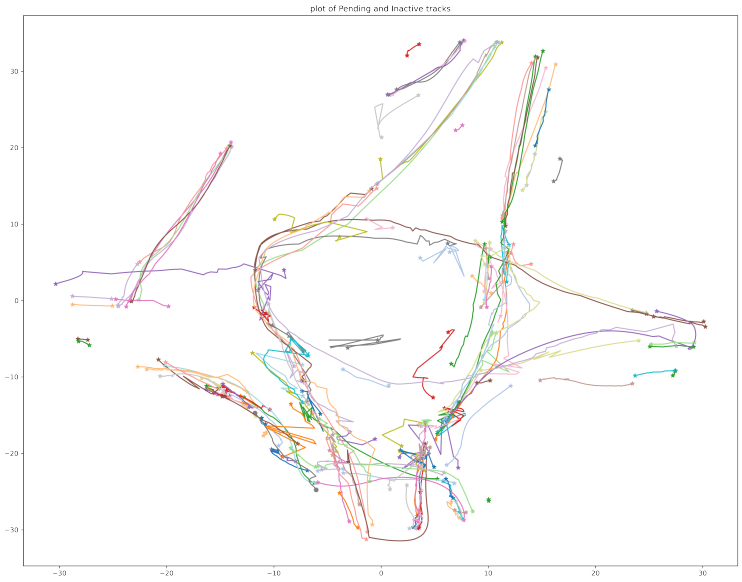


Figure 4: All the *Inactive* and *Pending* tracks detected by the tracker by the 740^{th} iteration

- *max pending time*: how long the tracks remain pending.
- *remove time*: maximum length of a new track to discard it if entering *Pending* status.
- ω_1 and ω_2, weights for the mixed IoU-Euclidean distance. These weights have been set to 0.1 for $1 - d_{IoU}$ and 0.9 for d_{Eucl} respectively. The weights need not sum to 1, although in principle the distance can be scaled by any multiplicative factor, if adjusting properly *max dist threshold*. The process of tuning these weights appears particularly complicated, as overfitting - underfitting tradeoffs may occur at different distances for different classes of objects and even in different parts of the image.

In general, tuning was performed by visual observation of tracks, attempting to weigh model flexibility and noise removal, in a similar fashion to the machine learning trade-off between underfitting and overfitting. As such, we believe the process is highly context-specific, with many influential factors such as, for example, the positioning of cameras, where objects appear and which types of objects are detected. Thus, careful tuning is suggested in every new tracking environment.

5 Results

In this section we will describe the analysis of final tracker results, first on an aggregate basis, then by examining specific relevant cases and finally comparing the tracks to the offline benchmark which was provided together with the dataset. Unless otherwise specified, individual track plots provided will refer to the last frame in which the object was detected, and the most accurate sensor for this detection as from credibility regions introduced in 3.2, together with the previously detected positions (pink dots).

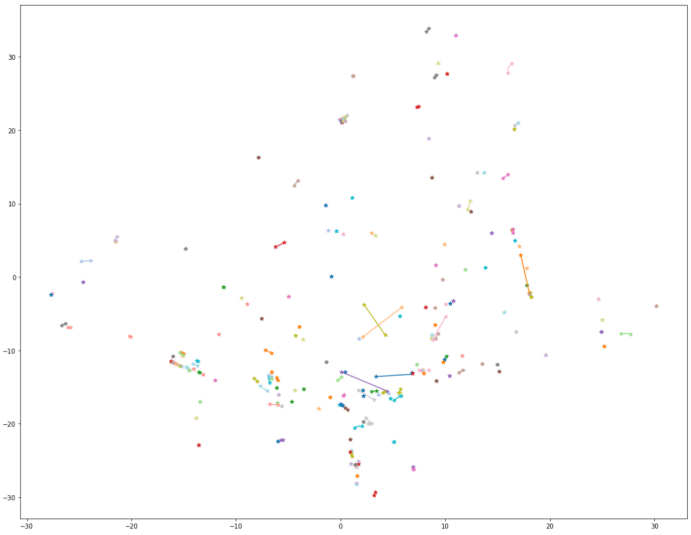


Figure 5: All *Removed* tracks detected by the tracker by the 740^{th} iteration

5.1 Aggregate results

In figure 4 we report the set of tracks that were considered *Pending* or *Inactive* by the end of tracking. The underlying shape of the urban landscape is clearly defined by the tracks, despite some noise. It can also be noticed that most observations from key areas of noise and uncertainty, such as the middle of the roundabout, have been removed thanks to the addition of credibility levels. Figure 5 reports the *Removed* tracks during tracking. It can be seen that most are individual detections, and a concentration appears in the lower part of the roundabout.

By the 740^{th} frame, the track counts were as follows:

New	Active	Pending	Inactive	Removed
2	1	1	210	152

Table 1: Track types by the 740^{th} frame

In the provided offline tracks dataset, 137 tracks were present. The maximum length of track detected with our algorithm was 198 frames, while for the offline benchmark it was 498, still with a 75% quantile at 157. This may be explained by the offline tracking being able to look back in time and hence make more accurate associations. In particular, the offline tracker is checking whether two separate inactive tracks can belong to the same object and merging these if this is the case.

5.2 Individual cases

Figure 6 reports a first example of track. The vehicle is fully tracked, with a quite smooth track profile. In particular, velocity and acceleration plots are also reported. Blue points correspond to active status, while in red are reported frames in which Kalman predictions were unsupported by detections, that is when some obstruction or double detection caused the track to become pending. Accelerations appear quite noisy in the beginning, due to accumulating error in the second differences, but taking the overall average allows to adjust the starting point for the filter, which quickly converges. The track becomes pending at least 3 times, but it is able to recover after few iterations. Discontinuity in velocities may be observed around those points, since when the track becomes active again, a correction of acceleration is needed. Indeed, the motion has non constant acceleration. Finally, it is quite interesting to observe the speed, which appears to oscillate significantly. It is unclear whether the observed behavior is purely noise driven or it was the actual motion.

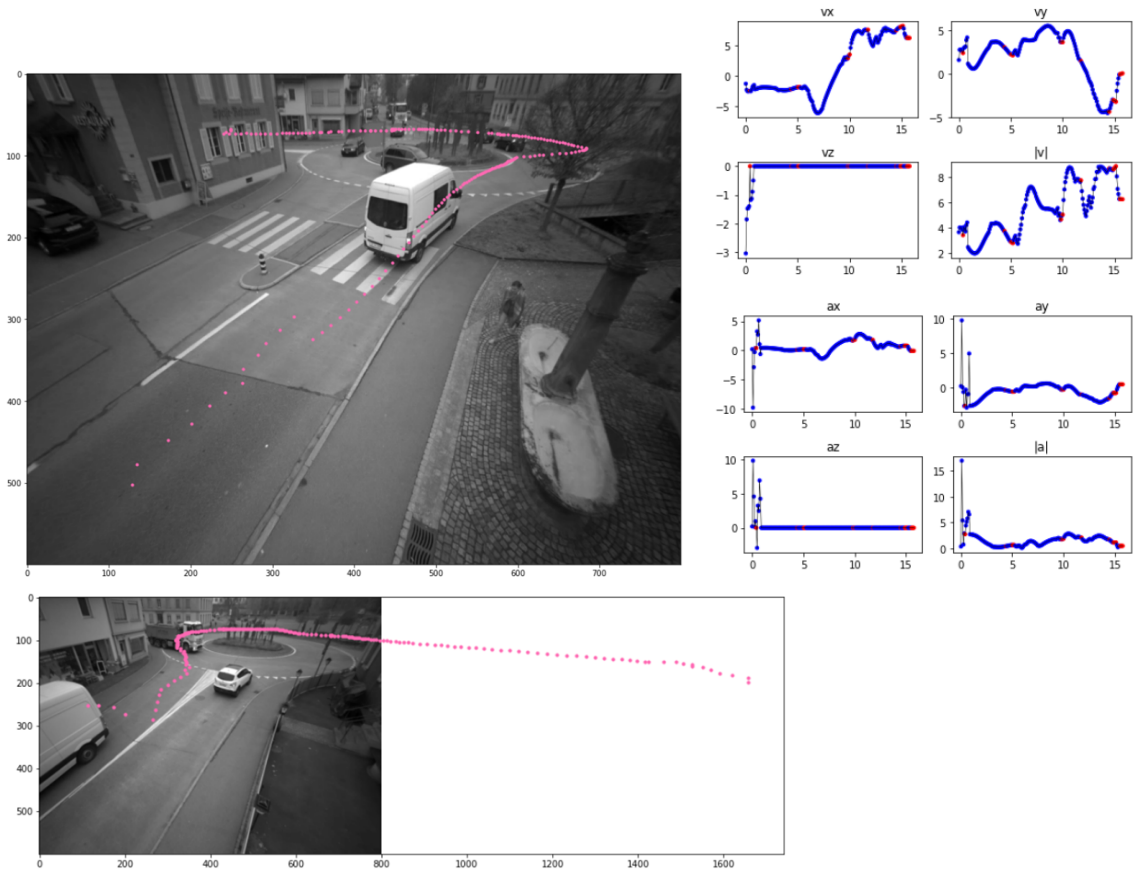


Figure 6: The trajectory of a van with velocity and acceleration profiles

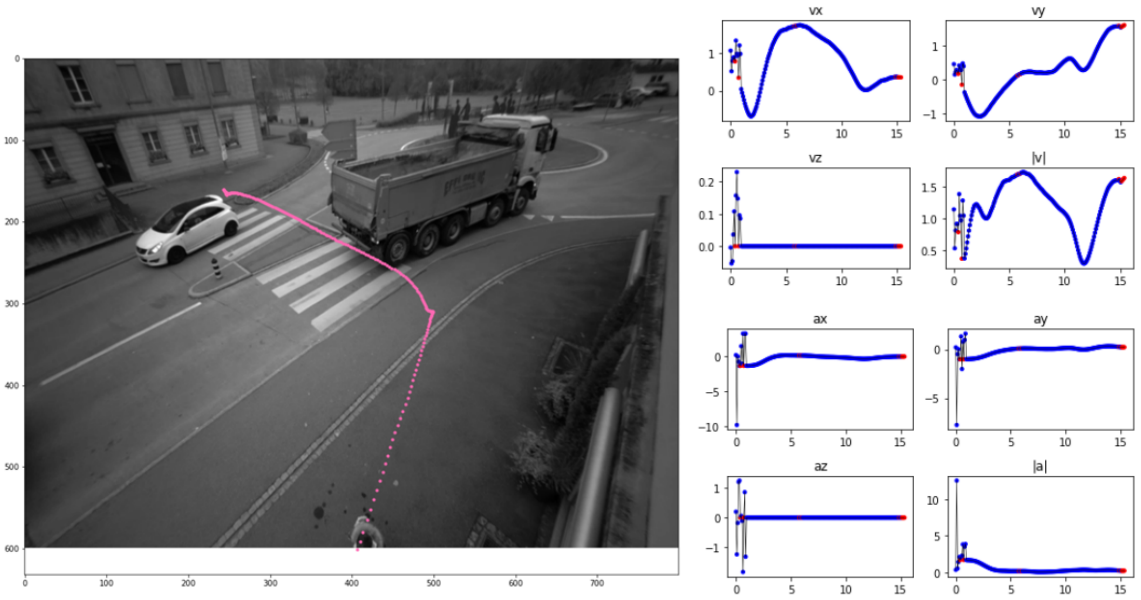


Figure 7: The trajectory of a pedestrian with velocity and acceleration profiles

The latter hypothesis is at least plausible, since the van has three points of likely decrease of speed (the first to give way to a car when entering the roundabout, the second when the same car slows down to exit the roundabout and the van has to slow down as well, and the third when going out the roundabout). However, further investigation by matching the frames with the observed slower speeds would be required to clarify the issue.

Figure 7 represents the movement of a pedestrian crossing the road. When dealing with pedestrians or small objects like bikes, crossing roads or walking on the sidewalk, the tracks are usually smoother and compact. Here the predictability of movement is greater, even though for non linear movement the task is more challenging. Acceleration is nearly zero throughout the track, which makes it adequate for a constant acceleration model, and the overall magnitude of speed is coherent with the movement of a pedestrian. Furthermore, it can be seen how the velocity in the vertical direction v_z was put to zero to avoid introducing additional noise into the system without benefits. The same was applied to all tracks.

Arguably, large vehicles represent the most challenging type to track, due to the difficulty of obtaining one unique point for their detections. Multiple detections are frequent, and even when there is only one detection, this is likely far from another sensor's detection of the same object. To mitigate the problem, however, appearance might be used. See, for example, [7]. In the considered scene of Figure 8, the three sensors' detections originated three independent but overlapping tracks. A method to tackle this type of situations is the merging algorithm explained in 6.

Conversely, it may also happen that close observations of different objects are swiftly joined together. This is exemplified in Figure 9, with a car-car association. In other cases, this may happen for pedestrian-car pairs or pedestrian-pedestrian pairs. The car in figure entered the view when another car was exiting, and the trajectory was unified since the detections were close enough. It is barely noticed a discontinuity in the velocity.

For further investigation of the algorithm's performance, in Appendix A additional interesting cases were reported.

Figure 8: Three tracks related to the same large vehicle

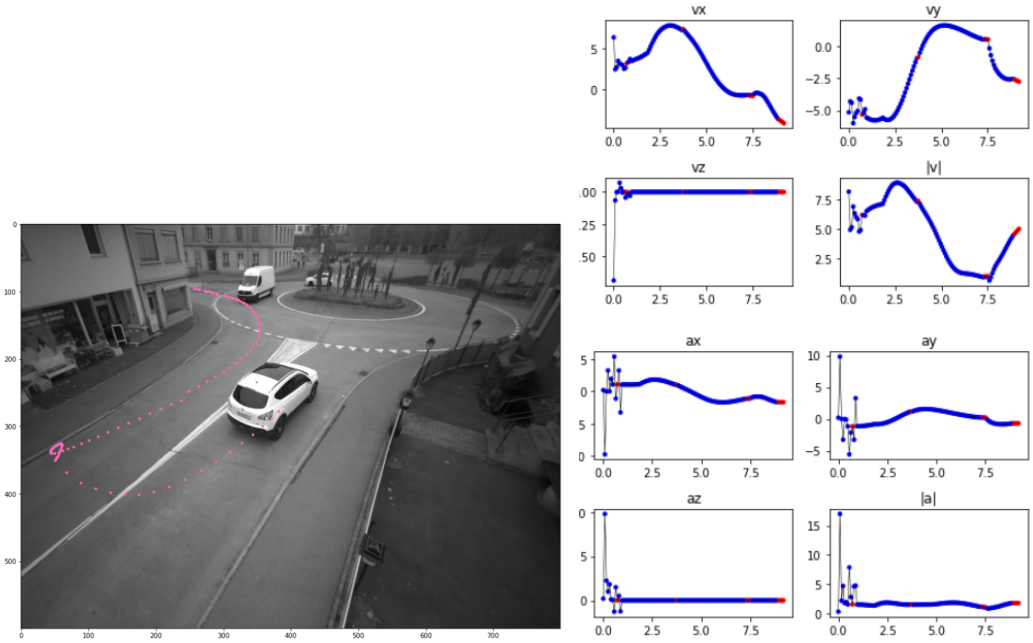


Figure 9: Joined trajectory of two different cars

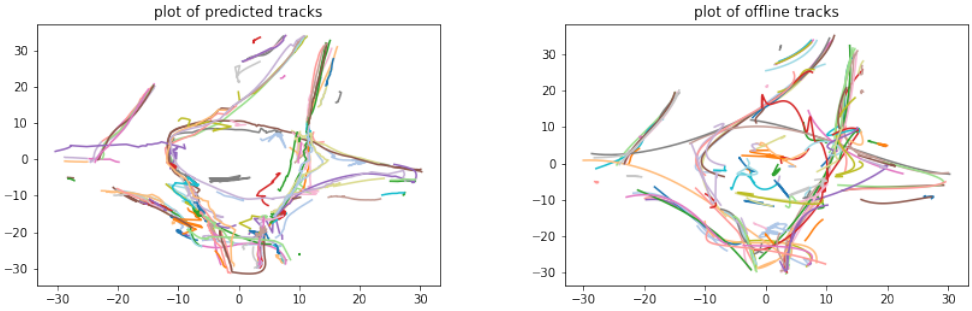
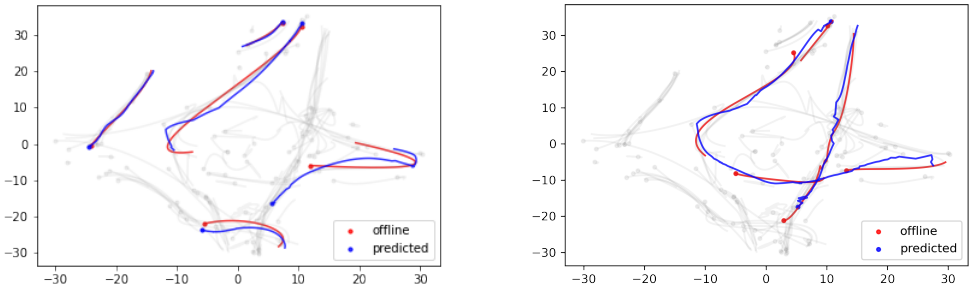


Figure 10: Top view for the tracks produced by our online algorithm and an offline algorithm



(a) (b)

Figure 11: 1-to-1 comparisons for the tracks produced by our online algorithm and an offline algorithm

5.3 Comparison with benchmark

A qualitative comparison to the provided offline tracks (Figure 10) shows sufficient ability to replicate the benchmark, although all tracks appear overall more noisy. It can be observed our algorithm is generally able to remove noise around critical zones of occlusion better, such as in the middle of the roundabout, thanks to the introduction of credibility levels.

Some one-to-one comparisons may be observed in Figure 11. It can be assessed that vehicle trajectories suffer from noise, while pedestrian trajectories achieve more satisfactory smoothness levels (these are the short trajectories on the top, bottom and left in Figure 11a). Both online and offline algorithms introduce an error in the *U-Turn* example reported in Figure 9, visible on the right side of Figure 11a. Finally, some longer tracks were reported in Figure 11b, including the trajectory of Figure 6. It can be noticed a fragmentation of the offline tracks for the same trajectory, probably due to measurement noise in the intersection viewpoints among sensors.

6 Track Merging Algorithm

Given the result of the presented Tracking algorithm, it is evident that one of the key area of improvement consists in developing an algorithm able to detect as much as possible tracks related to the same object and merge them. Such an instrument would be particularly useful in the case of large vehicles tracking, where the dimension of the object usually leads to multiple detections, describing different parts of the vehicle. In this case it is common that the tracker creates more than one track, with many of those being short and abruptly stopped as the detection ceases to exist.

It is important to notice that the merging algorithm presented in this section has been developed

as an offline instruments for reasons of time, and is thus employed only after the tracker has completed his process. A better use of this work would imply the integration of the code inside the tracker itself, maybe activating it every some time-steps. In this way it would be possible to create more stable and long active tracks, reducing the noise during the online processing and fostering the convergence of the Kalman Filter.

6.1 Algorithm

The starting point of the algorithm consist in the observation that often tracks representing the same objects present significant overlaps, and that it could be possible to exploit them for identification. Specifically, all the tracks are propagated for 20 time frames, using the relative Kalman Filter for prediction. We choose to remove all the tracks moving with a mean velocity smaller than 2 m/s, in order to avoid considering pedestrians, keeping only cars, cyclists and scooters: this is due to the fact that this kind of vehicles have in general more consistent and predictable tracks, and are consequently easier to merge. Moreover, pedestrians tend to move in groups, making them more difficult to distinguish for the algorithm.

Given a pair of tracks, for each frame the difference in velocity and positions are computed, both as a vector and in with their absolute value. The angle between the two trajectories is also computed for every instant, using the scalar product of velocities.

A merging between the two trajectories is then proposed only if the mean difference in position is smaller than 7.2 meters, the mean difference in velocity is smaller than 13 m/s and the mean normalized scalar product is greater than the empirical threshold of 0.3.

It is relevant to notice that these thresholds have been chosen after observation of the proposed merges, and therefore could be improved or fine-tuned.

6.2 Results

The algorithm proposes 62 mergings out of 214 total non removed tracks, among which we provide some examples.

In Figure 12 we can see that the tracker created three main tracks representing the movement of the grey car, and a fourth very short track probably generated by a double detection: the merging algorithm recognises this and proposes a definitive merging.

The same happens in Figure 13, where the truck is recognized among multiple tracks, an example of the important impact that an effective merging algorithm can have in the specific case of large trucks.

7 Conclusions and future work

In this section we summarize the strengths of the algorithm, we suggest possible solutions to the critical aspects emerged from testing and propose key areas of improvement for a robust tracker.

7.1 Achievements

The final version of the algorithm's key strengths are the ability to filter out noise, the robustness induced by a combined distance, a mechanism for addressing track fragmentation in presence of overlaps and realistic performance on ordinary-size vehicles.
Filtering noise is derived from the credibility levels adopted, as well as the use of Kalman Filter. It was possible to observe concrete (but qualitative) results by adjusting the credibility areas and the Kalman Filter noise covariance matrices. The overall smoothness is still not comparable to the offline tracks profile, which however has an advantage in observing all points prior to smoothing the track (numerous smoothing techniques are then available, e.g. B-spline smoothing).
The presence of a mixed distance allows to keep tracking, in principle, when one distance is not available or not reliable. If correctly tuned, the weights may allow to intensify the influence of one distance over the other, possibly enhancing the accuracy of track-detection association without

Figure 12: Four tracks whose merge is proposed by the algorithm

Figure 13: Large truck whose merge is proposed

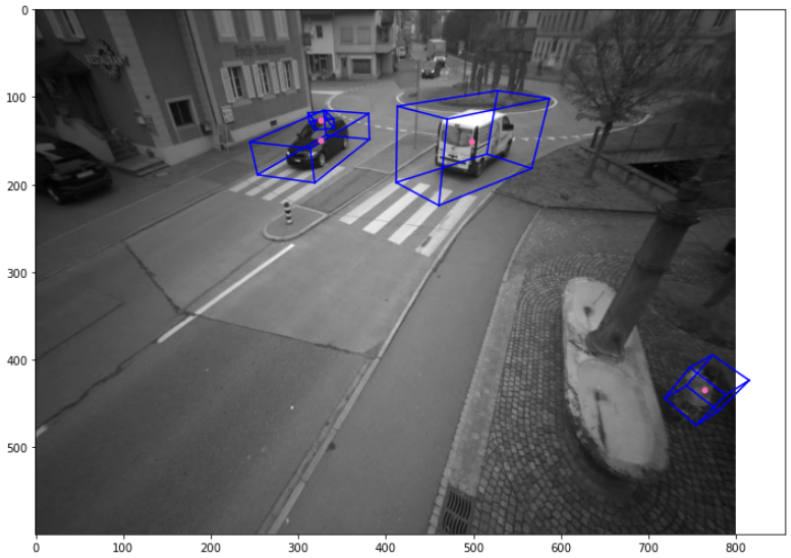


Figure 14: Example of double detection

compromising the process of filtering out the noise.
It has been noticed that, despite efforts in unifying same-object detections from different sensors, a tendency to instantiate multiple tracks for one object still appears due to measurement errors in the detections, even in quite regular cases. The merging algorithm aims at tackling those situations, providing a reliable tool for unifying parallel tracks. After processing the tracks with the algorithm the track list for ordinary vehicles is estimated to reduce of at least 1/3 in size.
The categories of vehicles which were observed to perform better in the tracker are pedestrians and ordinary-size vehicles. These are relatively smooth and the linear model generalizes well, given the limited velocities in the considered urban scenario.

7.2 Critical aspects and solutions

Some difficulties observed at testing time, which expose limitations of the current algorithm, are multiple detections, track-detection association in the context of many close objects and object mismatch, large vehicle tracking.

7.2.1 Multiple detections

When multiple detections are found for an individual object, which is not rare to observe for ordinary vehicles and very frequent for large-size vehicles, uncertainty is generated around the object in the subsequent frames. For better understanding the situation, let us consider the car on the left side of Figure 14. The car was previously tracked successfully until the point depicted in the figure, where a double detection occurred. As the association process only considers the global nearest neighbor in terms of the distance introduced in Eq. (16), the track was linked to the point with the smallest box, while a new track was instantiated for the other point, instead of discarding it. Notice that the box of the associated point must have had a much smaller intersection over union with the previous box, given by the difference in the absolute dimension of the two boxes. However, because in our algorithm we chose to keep a higher weight of the Euclidean distance between track state and detection over IoU, the proximity of points prevailed. In the subsequent frame, only one detection was ordinarily extracted from the car, and the original (and longest) track became pending. This happened because the new track found itself now closest to the next detection, and therefore was associated first, leaving no close-by detection to associate to the original track. The situation repeated for a few more frames until, by chance, the projected pending track returned to

be the closest to the new detection, and it was able to finish the tracking until the car disappeared from view. However, no general rule guarantees this fortunate outcome.

When considering possible solutions to this issue, multiple hypothesis tracking is one very valid opportunity, although its limits will be discussed in Sec. 7.4. However, it might be possible to still address the problem with a classical approach by favoring the association with the longest track of very close-with-one-another detections. Yet, we speculate that tuning the distance threshold for this criterion would represent a very challenging, and case-specific task.

7.2.2 Object mismatch

In Figure 7 it was observed a pedestrian track starting by the crosswalk. The previous track was terminated around the same position some frames before. This phenomenon is typical of overcrowded situations when object mismatches occur. In particular, when objects stop abruptly, the Kalman prediction is projected forward since the decrease in speed is interpreted as noise. The track is subsequently linked to the closest detection of this mistaken estimate of the state, which often happens to be another object. Several frames can pass before the transitory object leaves the area, and by then the track might be eventually too far away from the original object to link to it again, or it could be already discarded as *Inactive*. The straightforward solution in this case is again multiple hypothesis tracking.

7.2.3 Large vehicle tracking

In the previously presented example of Figure 8, it was anticipated that multiple detections are very common for large vehicles, and additionally much noise is introduced by the difficulty to assign a point detection to a very large object. Indeed, different views of the object will likely produce points near the rear, center and front of the vehicle, which is quite far, considering the object dimensions. To improve the treatment of these cases, it might be used an *ad-hoc* set of rules upon recognition of the category of the vehicle. Neural networks are suggested for the task, as well as heuristic methods based on the dimensions of the bounding boxes.

Having identified the large objects, one may adjust thresholds for detection merging and smoothing parameters. Alternatively, for the merging of parallel tracks the merging algorithm has also shown good results.

7.3 Improvements

Having observed the performance of the algorithm on the provided dataset, four key areas of improvement for more robust tracking were identified:

- *improvement of the detection algorithm*: noise can appear both in the form of multiple detections or detections that are observed far from their theoretical position. Such randomness is very hard to remove and may cause significant failures. The static nature of the data collection process facilitates the development of heuristic strategies to de-noise detections. Exploring them appears to us a key priority for the success of the algorithm.

- *improvement of track-detection association through velocity information*: velocity is included in the Kalman Filter state of each trajectory, but currently not used during association with new detections. Including some form of continuity among the current track velocity and the proposed velocity with the new detection appears an easy and effective enhancement of the present technology. For instance, it might help addressing cases like the one of Figure 9, where the car abruptly changes direction. In practice, one could use the cosine distance between the Kalman state velocity and the velocity estimated from linking the current (or previous) state to the detection.

- *improvement of the distinction of road areas*: while the algorithm has a primitive logic for separating roads and sidewalks, more rules, both location-specific and general, could be enforced inside the tracker to ensure to cut to the minimum the non-realistic trajectories and

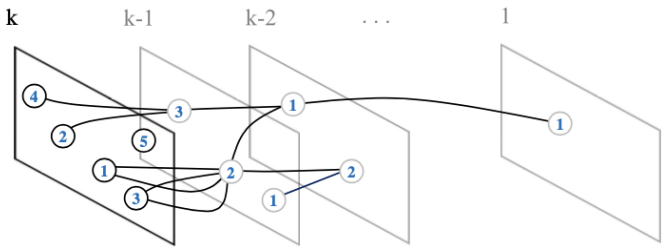


Figure 15: Subset of possible tracks hypotheses after the gating test at time k

providing aids to the track-detection association. For example, it would be useful to create areas of likelihood for the birth (and death) of a trajectory.

- *improvement of the motion model*: while a linear model is accurate for very regular classes of motion, more complexity was encountered even in the simple roundabout setting. Therefore, experimenting with alternative motion models could yield more desirable results. Some authors even suggest to run multiple models in parallel (cf. Sec. 8)

7.4 Multiple Hypotheses Tracking

Beside the possible improvements listed above, we want now to show another approach that might lead to better results in terms of detection-track association. We will just present the theory behind the basic Multiple Hypotheses Tracking (MHT), indeed we did not manage to accurately implement it ourselves for lack of time and for other reasons we will better point out afterwards.

7.4.1 Reasons behind this new approach

So far we have explored the conventional single hypothesis approach, described as visiting algorithm in 4.5, a global nearest neighbor algorithm which only considers the single most likely hypothesis for track update, where 'most likely' is defined in terms of the distance introduced . In this case, unassigned observations initiate new tracks and the associations are done under the constraint that an observation can be associated with at most one track.

This single hypothesis strategy works well in the case of widely spaced targets, accurate measurements, and few detections at low distance from the predicted next step of the tracks. In case those conditions do not hold, it's likely to have a mis-association (a false alarm chosen instead of the correct target) that leads to the loss of the real track. Moreover, if a detection is not associated, a new track starts and few different tracks with close by detections tend to mix between each other the associations of the correct ones.

Multiple hypotheses tracking algorithm is proposed to deal with these problems in the data association.

7.4.2 Algorithm

Imagine we have a given frame k where the detections from the 4 different sensors have already been joined if they represent the same object from different points of view. In this new approach alternative data association hypotheses are formed whenever observation-to-track conflict situations occur. Then, rather than choosing the best hypothesis, the hypotheses are propagated into the future in anticipation that subsequent data will resolve the uncertainty. Multiple Hypothesis Tracking maintains multiple track trees, and each tree represents all the hypotheses that originate from a single observation.

In figure 15, we can see a subset of the possible tracks originated from the only detection at frame 1 and from the non-associated detections in the next frames. The corresponding trees are the ones in figure 16. As we can see from figure 17, at each frame k a gating area is predicted for

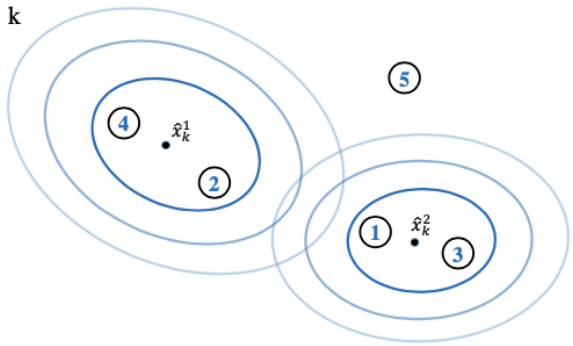


Figure 16: Example of gating areas for two tracks hyptotheses. $\hat{x}_k^l$ represents the likely location of the l^{th} track at time t

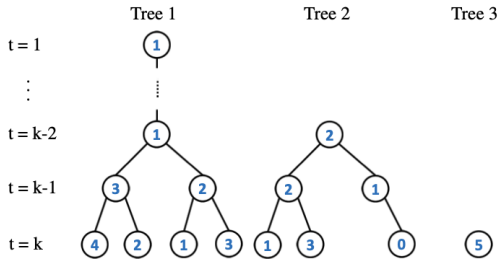


Figure 17: The track trees corresponding to tracks in 15. Each tree node is associated with an observation in the frame t

each track hypothesis which specifies where the next observation of the track is expected to appear. This area, represented with circles, is connected to the Mahalanobis distance between the location of the detections and the predicted location. Each track hypothesis is extended by appending new observations located within its gating area as its children and each new observation spawns a separate branch.

At each frame, each track hypothesis is associated with a track score depending on the motion model. It expresses whether a track hypothesis is more likely to be a true target or false alarm. We want now to determine the most likely combination of compatible object tracks (called 'global hypothesis') through solving an optimization NP-hard problem that maximizes the total track score. This problem can be represented in figure 18 through an undirected graph in which each track hypothesis is a node and an edge connects two tracks that are conflicting, not compatible.

Multiple Hypothesis Tracking has the potential to explore the solution space exhaustively, but has traditionally been slowed down by the exponential growth in the number of hypotheses. That's why its performance strongly depends on the ability to prune branches in the search tree quickly and reliably, in order to keep the number of track hypotheses manageable.

An example of a pruning strategy is the N−scan pruning, applied in figure 19 with $N = 2$: we first identify the tree branches containing the object tracks that solved the optimization problem. Then for each of the selected branches, we trace back to the node at frame kN and prune the subtrees that diverge from the selected branch at that node. After pruning, track trees that do not contain any track in the global hypothesis will be deleted. Besides N−scan pruning, we also prune track trees that have grown too large.

Figure 20 represents the track hypotheses that are left after the pruning.

To conclude, the steps described are iterated for each frame until the final one we have available.

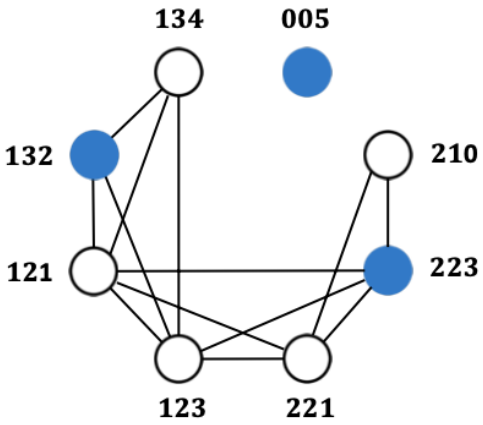


Figure 18: An undirected graph in which each track hypothesis is a node and an edge connects two tracks that are conflicting

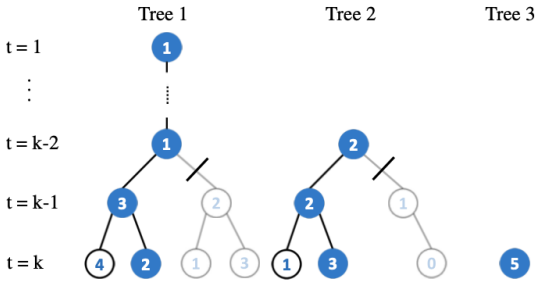


Figure 19: N-scan pruning example (N = 2): at frame k-2 we prune the subtrees that diverge from the selected branch at that node

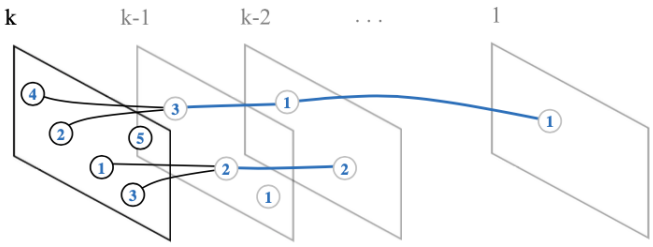


Figure 20: Track hypotheses after the pruning. The trajectories in blue represent the finalized measurement associations

7.4.3 MHT revisited

From those theoretical considerations it's clear how real-time applications of the algorithm are precluded by its time and space complexity. Some tricks should be then used to make it more practical.

- Apart from considering the motion model assumptions in the track score, taking the object appearance information into account is essential to improve the tracking algorithm: this information is more effective than motion features in reducing the number of look-ahead frames that are required to resolve data association ambiguities.
- To handle the issue of solving an NP-hard optimization problem at each frame, there is an online scheme which formulates the problem using feedback information from the previous frame's result to find optimal tracks. This will significantly reduce the computation effort improving performance.
- The combination of MHT with group tracking can be useful when there are time frames in which the proliferation of closely spaced targets may cause the number of hypotheses formed to become prohibitive. Group tracking is used until the targets separate sufficiently to allow feasible MHT of individual targets.

8 Further readings

Alternative approaches to our pipeline might be employed. In [6] a reinforcement learning approach is used starting from off-line ground truths. Many authors also include appearance models to improve the understanding of objects in the environment. The approach consists in describing objects through features (such as the encoding obtained through neural network autoencoders) and associating subsequent objects based on the cosine distance among the feature vectors. See, for example, [7] and [4].

Moreover, [3] and [8] go deeper into the alternative Multiple Hypotheses Tracking implementations and variants, considering also the ones mentioned in 7.4.3.

To see further achievements on the MHT approach, [1] explains how it can be efficiently combined with multiple model filtering. Here the state estimates and the covariance matrices of the multiple Kalman filters are combined via the process defined to be mixing. The basic principle is that the currently more accurate (as determined by the computed model probabilities) models transfer their state estimates to the less accurate models.

9 References

[1] Samuel S Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, 2004.

[2] 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.

[3] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 4696–4704, 2015.

[4] Weiqiang Li, Jiatong Mu, and Guizhong Liu. Multiple object tracking with motion and appearance cues, 2019.

[5] 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.

[6] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4705–4713, 2015.

[7] Min Yang and Yunde Jia. Temporal dynamic appearance modeling for online multi-person tracking. *Computer Vision and Image Understanding*, 153:16–28, Dec 2016.

[8] Haanju Yoo, Kikyung Kim, Moonsub Byeon, Younghan Jeon, and Jin Young Choi. Online scheme for multiple camera multiple target tracking based on multiple hypothesis tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(3):454–469, 2016.

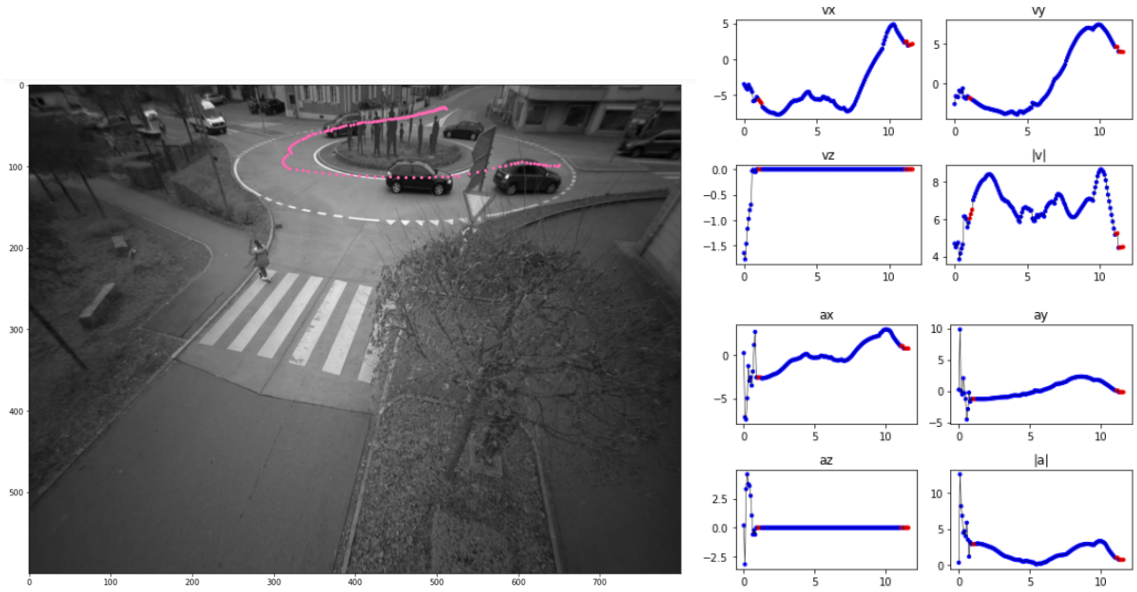


Figure 21: Additional track 1

A Additional images

In this section we report compelling pairs of images for trajectory and speed/ acceleration plots.

Figure 22: Additional track 2

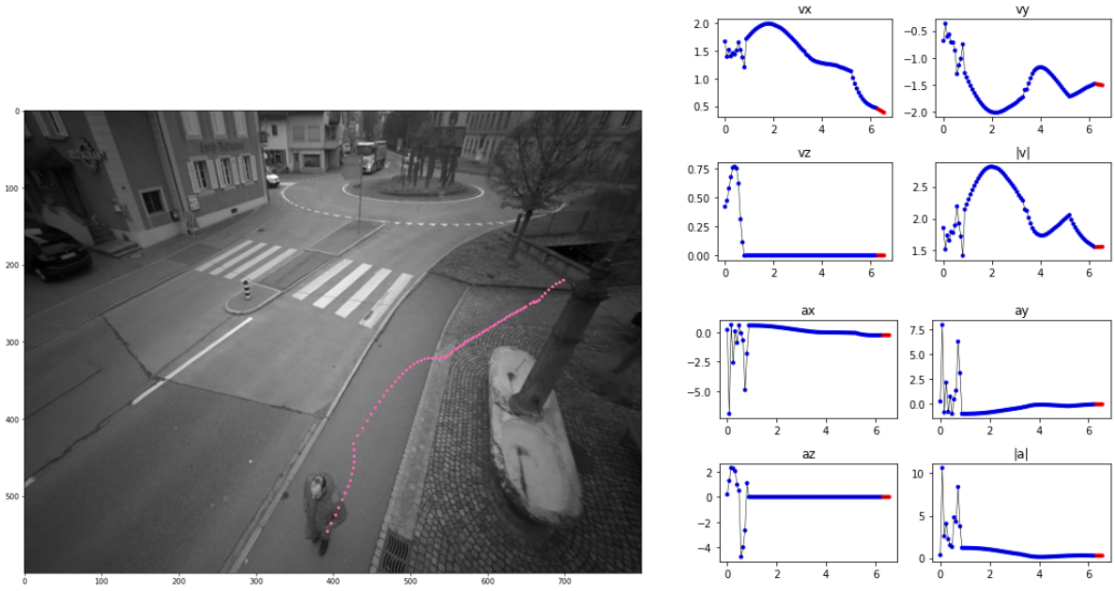


Figure 23: Additional track 3

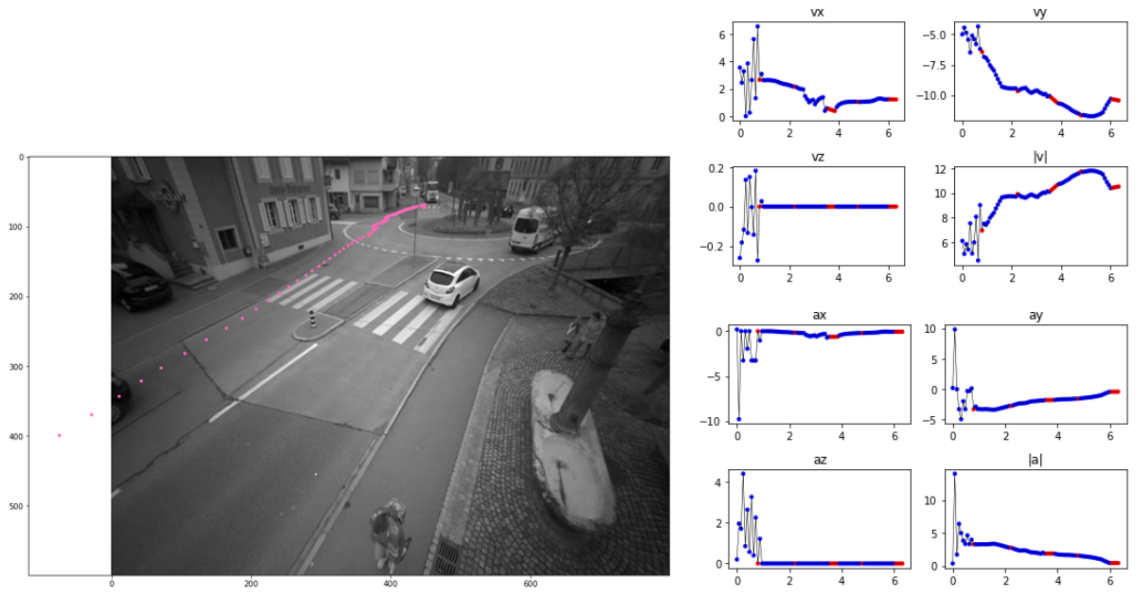


Figure 24: Additional track 4

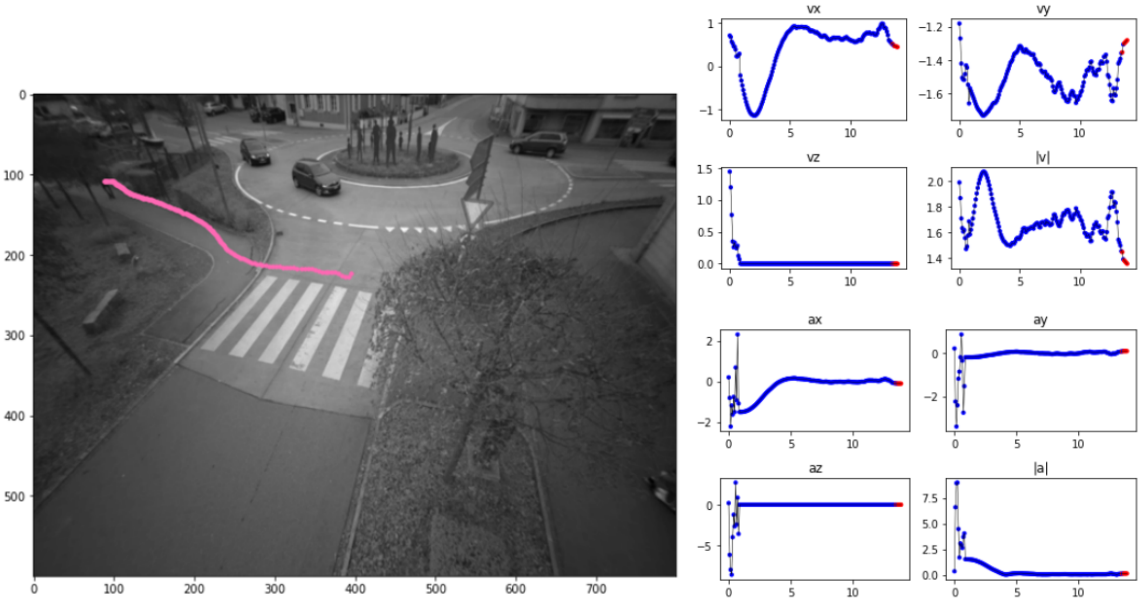


Figure 25: Additional track 5

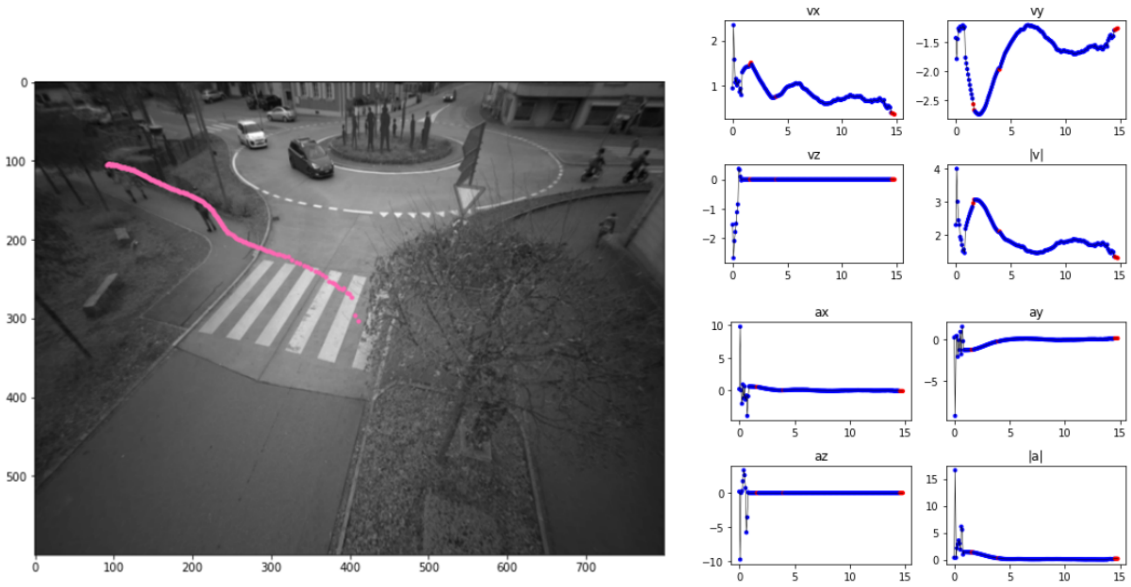


Figure 26: Additional track 6

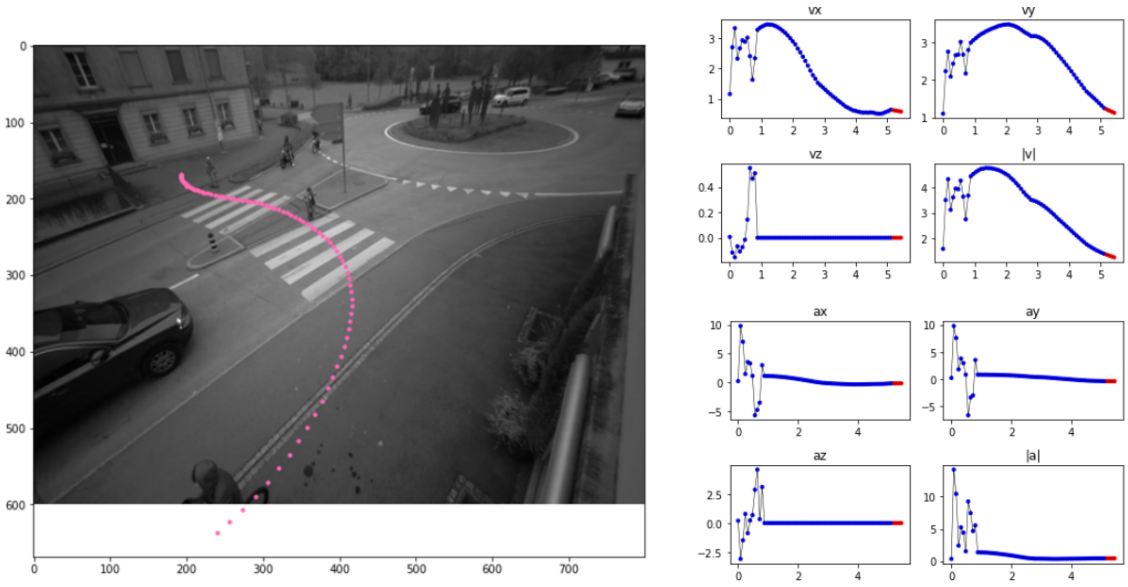


Figure 27: Additional track 7

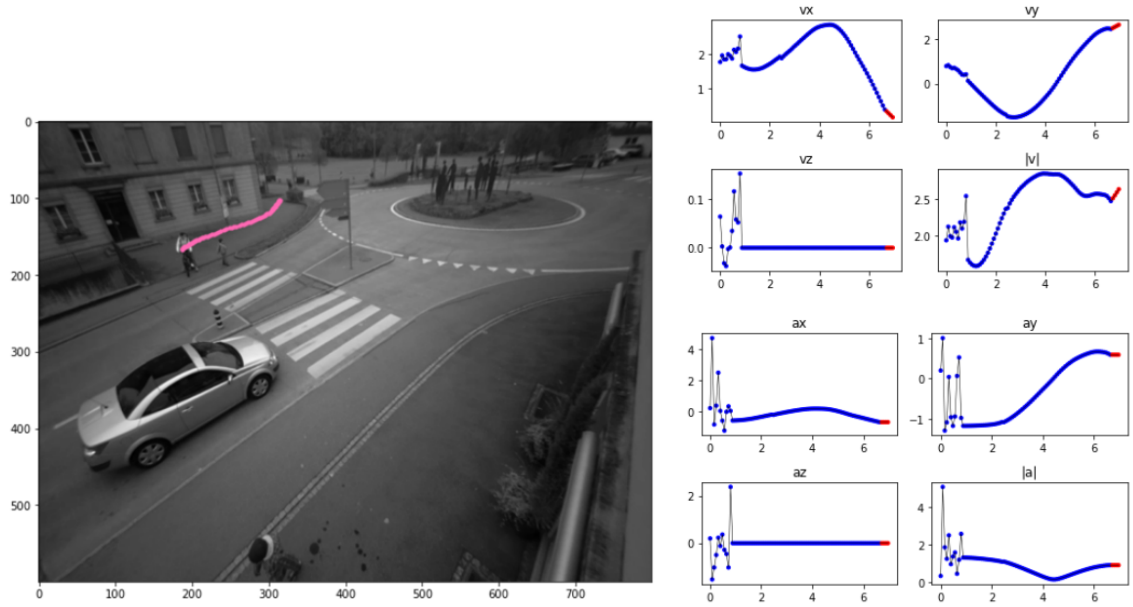


Figure 28: Additional track 8

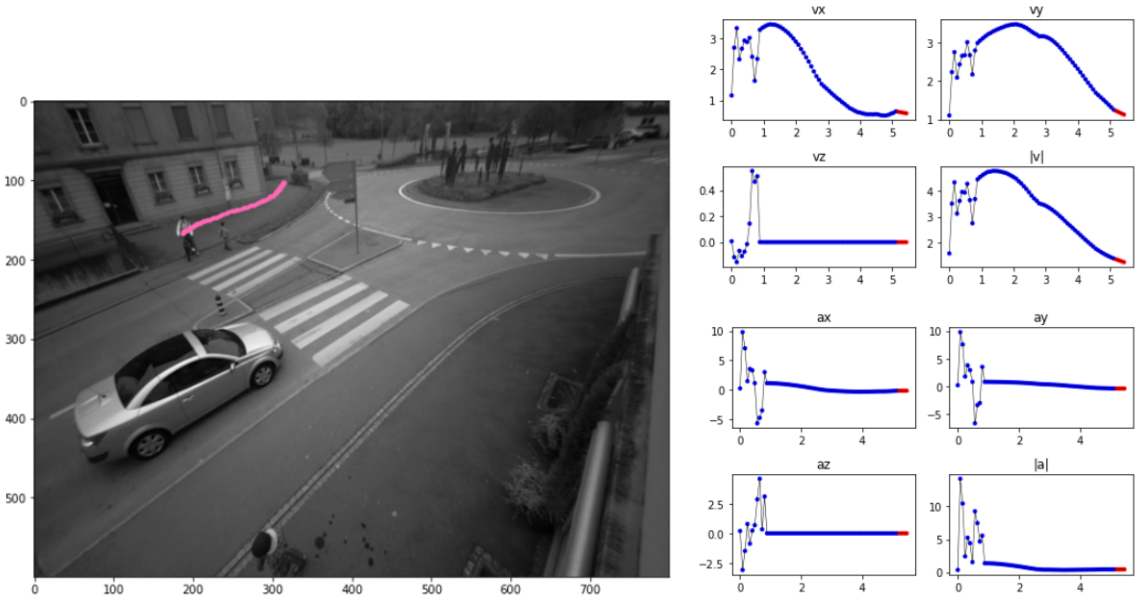


Figure 29: Additional track 9

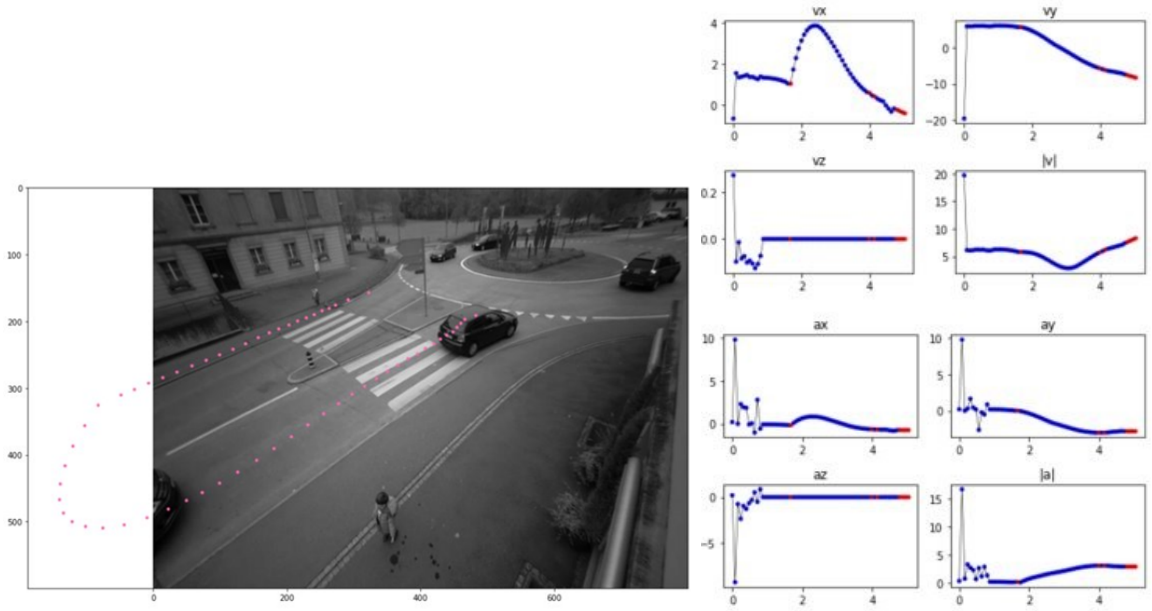


Figure 30: Additional track 10