Enhanced Road User Tracking in Cluttered Urban Environments through Multiple Hypothesis Tracking and Probabilistic Object Discrimination

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Author: **Francesco Romeo**

Student ID: 968385
Advisor: Prof. Simone Vantini
Co-advisors: Yury Tarakanov
Academic Year: 2021-22

Abstract

The focus of this thesis is to develop a tracking system, based on the Multiple Hypothesis Tracking algorithm, that is able to build trajectories of road users in complex urban scenarios, i.e., scenarios involving pedestrians, cyclists and vehicles. In particular, this work analyzes the performance of such a tracking system when coupled with a strategy to address the problem of multiple road users fused into a single detection. The multiple hypothesis approach consists of building different hypothetical trajectories for the same road user and deciding which one is the most likely to be the real one only after having analyzed how these hypothetical tracks evolve in future frames. All tracking systems face well-known challenges: sensor noise, occlusions of objects and detections that are not representing one target but two or even no real target at all. The last issue has been directly tackled in this work by means of a strategy to detect fused objects, i.e., cases of single detections merging multiple road users. The strategy in question consists of building a bootstrap confidence interval for the volume of the target based on detections associated with the correspondent track. This confidence interval can help in identifying detections that are likely to not come from the same target and that might represent fusion with another nearby road user. It is worth mentioning that the tracking system uses a Kalman Filter to build the trajectories of the targets.

The tracking system has been tested on a dataframe containing 750 frames. Each frame contains information on the position of the targets in the frame and on their appearance through a bounding box. The algorithm can solve all cases of fused detections and provides qualitatively good tracks. The main challenges that have not been solved yet are the presence of tracks that do not correspond to real targets and are generated by noisy detections, as well as the interruption of consolidated tracks as a result of the presence of obstacles that occlude the objects.

Keywords: Multiple Hypothesis Tracking, Bootstrap Confidence Intervals, Tracking System, Merged Objects, Kalman Filter

Abstract in lingua italiana

Questa tesi è incentrata sullo sviluppo di un sistema di tracciamento, basato sull'algoritmo Multiple Hypothesis Tracking, in grado di tracciare oggetti in scenari urbani complessi, ovvero scenari che coinvolgono pedoni, ciclisti e veicoli. In particolare, questo lavoro analizza le prestazioni di tale sistema di tracciamento accoppiato con una strategia che vuole risolvere il problema della presenza di detections che includono più oggetti contemporaneamente. L'approccio Multiple Hypothesis consiste nel costruire diverse ipotetiche traiettorie per uno stesso utente al fine di decidere qual è la più verosimile solamente dopo aver analizzato come queste traiettorie ipotetiche evolvono nei frames successivi, guardando le loro caratteristiche di moto e di aspetto. Un sistema di tracciamento deve sempre affrontare alcune sfide ben note: rumore nei dati, occlusioni di oggetti e detections che non rappresentano un solo utente, ma due o addirittura nessuno. L'ultimo problema è stato affrontato in questo lavoro mediante l'utilizzo di una strategia per rilevare oggetti che sono stati fusi in un unica detections. La strategia in questione consiste nel costruire un intervallo di confidenza per il volume dell'oggetto tracciato basato sulle detections precedenti, che può aiutare a identificare detections che probabilmente non provengono dallo stesso oggetto e che potrebbero fare riferimento a più oggetti uniti in un unica detection. È inoltre importante menzionare che il sistema di tracciamento utilizza un Kalman Filter per costruire le traiettorie degli oggetti presenti nello scenario analizzato.

Il sistema di tracciamento è stato testato su un dataframe formato da 750 frames. Ogni fotogramma contiene informazioni sulla posizione degli oggetti e sul loro aspetto, quest'ultimo rappresentato da una bounding box. L'algoritmo è in grado di risolvere tutti i casi di detections contenenti più di un oggetto e fornisce traiettorie qualitativamente buone. I principali problemi che non sembrano essere risolti sono la presenza di traiettorie, generate da dati rumorosi, che non corrispondono a oggetti reali, ma anche l'interruzione di traiettorie consolidate causata dalla presenza di ostacoli che occludono gli oggetti.

Parole chiave: Multiple Hypothesis Tracking, Intervalli di Confidenza di Bootstrap, Oggetti Fusi, Kalman Filter

Contents

Introduction

Object tracking is the process of following moving entities in subsequent observations coming from one or multiple sensors, with the aim of estimating and predicting their trajectories. 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-world context, including sensor noise, occlusions and the presence of merged objects.

In this project the usage of Multiple Hypothesis Tracking (MHT) in complex urban scenarios will be explored and a method to detect cases of fused objects will be designed. In particular, the resulting tracker will be tested on a data set that comprises a collection of frames coming from one stereovision sensor designed by Viscando AB, a Swedish company specializing in traffic data collection and analysis for safe and smart mobility applications, in collaboration with which this work has been produced. The depicted scenario is a roundabout, and the objects that will be tracked represent any type of road user: pedestrians, cyclists and vehicles.

The aim of this thesis is to understand how MHT can help in case studies in which there are not only pedestrians but cars and cyclists as well, and to design and implement a strategy that is able to detect cases of multiple road users fused into one detection by the road sensors. The assumption of this work is that the combination of Multiple Hypothesis Tracking and this strategy will generate an algorithm that will be able to address the common problems that a tracking system faces and solve them, outperforming the conventional single hypothesis tracker.

It is worth mentioning that, alongside with MHT, a Kalman Filter will be used to process data representing the detections of the targets. Each track will have its own filter, whose role is to filter out the noise from the data and predict the next position of each track. The performance of the tracker will also be tested by using different motion models and studying how suitable they are for MHT, as well as by introducing some kinematic constraints, whose effects on the entire tracking system will be analyzed.

The thesis is organized as follows:

- Chapter 1 introduces the theoretical framework in which the proposed algorithm will operate. All the assumptions and computations related to Multiple Hypothesis Tracking will be described, along with an explanation about of Bootstrap Confidence Intervals, used to detect fused objects, are computed. This has to be considered the reference chapter for the implementation of the algorithm.
- Chapter 2 describes the data set that will be used to test the proposed algorithm.
- Chapter 3 shows how the algorithm has been implemented, wrapping up what is explained in Chapter 1 about MHT.
- Chapter 4 lists the results produced by the proposed algorithm on the given data set.
- Chapter 5 analyzes in detail how kinematic constraints affect the output of a tracking algorithm and the advantages and disadvantages of using different motion models.
- Chapter 6 draws the conclusions of this thesis and proposes further improvements for tracking in urban settings.

1 | Theoretical Framework

This chapter describes the theory at the basis of the proposed algorithm by first introducing Kalman Filters and then by explaning how Multiple Hypothesis Tracking (MHT) works together with other components of the algorithm. As a consequence, this chapter is thought of as a complete description of the assumptions and properties that will be used in the process of building the road users' trajectories.

1.1. Kalman Filter

The Kalman Filter is a recursive linear estimator that 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:

$$\begin{cases} x(t+1) = Fx(t) + Gu(t) + v_1(t) \\ y(t) = Hx(t) + Du(t) + v_2(t) \end{cases} \tag{1.1}$$

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 v_1 is called **state noise** or equivalently **model noise**, and it accounts for internal noise and modelling errors related with the state equations; v_2 is called **output noise** or equivalently **measurement noise**, and it accounts for noise coming from the data.

In this work the effects of external inputs will be neglected by putting the matrices G and D equal to the null matrix, and the state variables will be the position, speed, and acceleration of the objects in a 3-dimensional 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:

$$\mathbb{E}[v_i(t)] = 0, \quad \mathbb{E}[v_i(t)v_i(t)^T] = V_i, \quad \mathbb{E}[v_i(t)v_i(t-\tau)^T] = 0 \ \ \forall t,\ \forall \tau \neq 0 \tag{1.2}$$

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

- The following relationship holds between v_1 and v_2:

$$\mathbb{E}[v_1(t)v_2(t-\tau)^T] = V_{12} = \begin{cases} 0 & \text{if } \tau \neq 0 \\ \text{can be non-zero} & \text{if } \tau = 0 \end{cases} \tag{1.3}$$

Due to the multiple advantages this filter has, the Kalman filter (KF) algorithm is often chosen as main feature for the implementation of a tracker. First of all, the KF requires no training data, which is convenient in this study case since there are not that 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

$$\hat{x}(t+1|t) = F\hat{x}(t|t-1) + K(t)e(t) \qquad \text{state equation} \tag{1.4}$$
$$\hat{y}(t|t-1) = H\hat{x}(t|t-1) \qquad \text{output equation} \tag{1.5}$$
$$e(t) = y(t) - \hat{y}(t|t-1) \qquad \text{output prediction error} \tag{1.6}$$
$$K(t) = \left(FP(t)H^{\mathrm{T}} + V_{12}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1} \qquad \text{gain of KF} \tag{1.7}$$

where $P(t)$ is the most important matrix of the KF together with the gain of the filter $K(t)$, and its expression is given by the **Difference Riccati Equation** (DRE):

$$P(t+1) = (FP(t)F^T + V_1) - (FP(t)H^T + V_{12})(HP(t)H^T + V_2)^{-1}(FP(t)H + V_{12})^T \tag{1.8}$$

In order to guarantee the existence of DRE at any time instant t, the matrix

$$HP(t)H^T + V_2$$

must be invertible. The quantity $HP(t)H^T$ is a symmetric and positive semidefinite matrix, thus one additional condition is required to guarantee the invertibility of the whole block: V_2 must be positive definite. It is common to make this assumption, in order to obtain the existence of the DRE at least from a theoretical point of view.

Equations (1.4) and (1.8) are dynamic equations and thus they need initial conditions:

$$\hat{x}(1|0) = \mathbb{E}[x(1)] = X_0 \qquad P(1) = \mathrm{Var}[x(1)] = P_0 \tag{1.9}$$

These two conditions form a probabilistic description of initial state $x(1)$. The initial condition of matrix $P(t)$ is linked to an important property of this matrix:

$$P(t) = \mathrm{Var}[x(t) - \hat{x}(t|t-1)] \tag{1.10}$$

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

A fundamental assumption to formally guarantee the optimality of KF is that $x(1)$ is uncorrelated with both white noises $v_1(t)$ and $v_2(t)$.

The equations above describe how it is possible to perform prediction tasks using KF, but, as stated above, KF is important also to filter out the noise from the data. The equations that characterise the filtering process, which allows to obtain the state vector $\hat{x}(t|t)$, are the following: at iteration time t

$$\hat{x}(t|t) = F\hat{x}(t-1|t-1) + K_0(t)e(t) \tag{1.11}$$

$$K_0(t) = \left(P(t)H^{\mathrm{T}}\right)\left(HP(t)H^{\mathrm{T}} + V_2\right)^{-1} \tag{1.12}$$

where $e(t)$ has the same definition as in equation (1.6). These equations are valid under the restrictive (but very common) assumption that $V_{12} = 0$.

1.2. Multiple Hypothesis Tracking

MHT was proposed for the first time by Reid in 1979 [12]. The main idea of this new algorithm is to create different hypotheses representing different trajectories for the same target. In particular, a hypothesis is formed by an existing trajectory and a detection, i.e., a data point, that could represent the next position of the target. In this way all the probable next positions of the track are considered and only the very unlikely ones are discarded. Each of the track hypotheses has a score which depends on the motion and appearance features of the collected detections. The tracks with the highest scores are more likely to represent an existing target and will thus be used to decide which other hypotheses to keep and which to discard.

It is already possible to understand that the great disadvantage of such an algorithm lies in its time complexity. The number of hypotheses that are created might eventually explode. This is why it is very important to control the number of hypotheses by deploying a good hypothesis reduction strategy.

In this work a track-oriented MHT will be used. This approach is inspired by [7] and differs from the one presented in [12], which is measurement-oriented. As stated in [12], in the measurement-oriented version every possible track is listed for each measurement, while the opposite holds in the track-oriented version. This means that, in the measurement-oriented version, one hypothesis is a list of tracks' ID indicating which track the *i-th* measurement is associated with, while in the track-oriented version an hypothesis is the list of measurements associated with one track.

As stated above, the main advantage of Multiple Hypothesis Tracking is that it considers multiple tracks for one target and eventually chooses the one that is most likely to represent the real trajectory of the target, also based on its compatibility with other tracks related to other targets. It will be clear that one of the most important steps of MHT is to identify a set of tracks that can coexist and that have the highest global score.

A convenient structure to represent the tracks at every time step is a **tree** structure, in which every node consists of an observation (i.e.,a detection coming from the data set) and every path from the root to a leaf represents a possible track. This structure is the natural result of the algorithm, given the fact that at every time step there is the possibility that a track is associated with more than one detection, and that each one of these associations is depicted in the tree by a new branch. This means that a tree will be a set of track hypotheses representing the same target. As a consequence, all the track hypotheses in one tree are **incompatible**, where this term is used to indicate that two or more track hypotheses share at least one measurement.

In the following subsections, the steps of MHT will be described in detail considering a generic time step k. It is worth remembering that the work in [7] inspired the theoretical framework for MHT.

1.2.1. Update of Hypotheses

It is important to recall what an hypothesis is: a collection of measurements that ideally are referred to the same target and that come from subsequent frames in the data set. Therefore, an hypothesis is a possible trajectory for a certain target.

Each hypothesis created up to the k-th generic time step has its own motion estimate. Consider a generic track hypothesis. An order between tracks does not exist, unless the score is used to order them, but for the sake of simplicity, this generic track will be called the *i-th*. The Kalman Filter keeps track of the positions that form this *i-th* hypothesis in order to estimate its speed and acceleration. The Kalman Filter can also predict the next position of the track hypothesis based on the data that has been associated with the track itself up to the current time step.

First of all, the next position of the *i-th* track hypothesis at time step k is assumed to be a gaussian random vector whose mean is the position predicted by the filter, indicated as $\hat{x}_k^i$, and whose covariance is the covariance of the filter, indicated as Σ_k^i:

$$X_k^i \sim \mathcal{N}(\underline{\hat{x}}_k^i, \Sigma_k^i)$$

Based on this information, it is possible to identify the area in which the next position of the track is expected to fall, which is called "**gating area**" of the track. The **gating area** of the track is given by the points whose squared Mahalanobis distance from the track's predicted position $\hat{x}_k^i$ is smaller than a fixed threshold:

$$d^2 = (\underline{x} - \underline{\hat{x}}_k^i)^T (\Sigma_k^i)^{-1} (\underline{x} - \underline{\hat{x}}_k^i) \leq d_{th} \qquad \underline{x} \in \mathbb{R}^3 \tag{1.13}$$

To be more specific, the *gating area* is an ellipsoid that is delimited by the level curve of the density function of the random vector X_k^i, corresponding to the unknown position of the track at time step k, and whose shape depends on the covariance of the filter. All the data points, called **detections**, that fall into this gating area are possible next positions of the track and will thus be associated with the track for updating purposes.

Each of these associations is represented in the tree of the *i-th* hypothesis as a new branch having as its leaf the correspondent detection. As a result, for each new association there is a new hypothesis in the tree, represented by the path from the root to one of the leaves. Moreover, each of the new associations will independently update the filter of the hypothesis that originated them.

For example, suppose that the *i-th* track hypothesis formed at step $k-1$ has three detections in its gating area. Each of these therefore may be its next position at step k. This will lead to three new branches starting from the leaf node that corresponds to the *i-th* track hypothesis in the tree to which the track belongs. As a consequence, there are three new paths from the root to the leaves, that is, three new hypotheses.

It can also happen that a track does not have any detection in its gating area. In this case, the track hypothesis is updated as if its detection is missing, meaning that its next position will coincide with the one predicted by the filter, i.e.,$\underline{\hat{x}}_k^i$.

It is worth mentioning that in [7] each detection starts a new tree even if it has been associated with a track. In the following implementation the scenario is not as crowded as an only-pedestrian one, and it is very unlikely that an object will appear in the middle of the scene. Moreover, following the aforementioned approach, the computational cost would become prohibitive. For these reasons, this step was not implemented in this work. As a consequence, in this implementation only detections that have not been associated with any track will start a new tree and a new hypothesis.

1.2.2. Tracks' Score

In order to be able to choose among all the tracks the ones that are most likely to represent the targets, each track should be scored based on the detections that it has been associated with in the previous frames. Considering for example the *i-th* track hypothesis, its score at frame k can be designed in the following way according to [7]:

$$S^i(k) = \omega_{mot} S^i_{mot}(k) + \omega_{app} S^i_{app}(k)$$

where S_{mot} and S_{app} are denominated **motion score** and **appearance score**, respectively, and should quantify how likely the motion feature and appearance feature of the track are to represent those of a real target. Their weights should be tuned based on the importance to be given to the two contributions. In this work they will be considered equally important so the weights will be set equal to one. By analyzing how these indices are computed it is possible to better understand their meaning.

The state-of-the-art in computing the **motion score** is summarized by S. Blackman and R. Popoli in [2]. It consists of computing the log-likelihood ratio related to the following statistical test:

$$H_0 : D_{i_{1:k}} \text{ comes from the background} \qquad H_1 : D_{i_{1:k}} \text{ comes from the track}$$

where $D_{i_{1:k}}$ is the set of observations that have been associated with the *i-th* track from time 1 to time k. Let $y_{i_{1:k}}$ be the set of measurements that correspond to the observations associated with the *i-th* track from time 1 to time k, then the *Log-Likelihood Ratio* (LLR) can be written as:

$$S^i_{mot}(k) = \ln\left(\frac{p(y_{i_{1:k}}|D_{i_{1:k}} \subseteq T_i)}{p(y_{i_{1:k}}|D_{i_{1:k}} \subseteq B)}\right)$$

where $D_{i_{1:k}} \subseteq T_i$ represent H_1 hypothesis and $D_{i_{1:k}} \subseteq B$ the null hypothesis. Assuming that under the null hypothesis measurements are conditionally independent, it is possible to rewrite the argument of the logarithm as follows:

$$\frac{p(y_{i_{1:k}}|D_{i_{1:k}} \subseteq T_i)}{p(y_{i_{1:k}}|D_{i_{1:k}} \subseteq B)} = \frac{\prod_{t=1}^{k} p(y_{i_t}|y_{i_{1:t-1}}, D_{i_{1:t}} \subseteq T_i)}{\prod_{t=1}^{k} p(y_{i_t}|D_{i_t} \subseteq B)}$$

with D_{i_t} representing the observation associated with the track at time t and y_{i_t} the correspondent measurement. It is possible to say that the distribution of the measurement y_{i_t} given the measurements associated in the previous time instants and the distribution of the position of the track at time t given the past measurements are the same, since they both refer to the position the track will have based on previous data. As a consequence, based on the assumption on the distribution of the future position of the track stated in the previous section, it is possible to say that:

$$p(y_{i_t}|y_{i_{1:t-1}}, D_{i_{1:t}} \subseteq T_i) = \mathcal{N}(y_{i_t}; \underline{\hat{x}}^i_t, \Sigma^i_t)$$

where $\underline{\hat{x}}^i_t$ is the position predicted by the Kalman Filter and Σ^i_t is the covariance matrix of the filter at time t.

Regarding the denominator, based on [2] the distribution of the measurement under the null hypothesis can be seen as uniform:

$$p(y_{i_t}|D_{i_t} \subseteq B) = \frac{1}{V}$$

where V is the measurement space, i.e.,the image area.

Thanks to the properties of the logarithm, a recursive formula for the computation of the motion score can be derived:

$$S^i_{mot}(k) = \ln\left(\frac{\prod_{t=1}^{k} p(y_{i_t}|y_{i_{1:t-1}}, D_{i_{1:t}} \subseteq T_i)}{\prod_{t=1}^{k} p(y_{i_t}|D_{i_t} \subseteq B)}\right) = \sum_{t=1}^{k} \ln\left(\frac{p(y_{i_t}|y_{i_{1:t-1}}, D_{i_{1:t}} \subseteq T_i)}{p(y_{i_t}|D_{i_t} \subseteq B)}\right) =$$

$$= \ln\left(\frac{p(y_{i_k}|y_{i_{1:k-1}}, D_{i_{1:k}} \subseteq T_i)}{p(y_{i_k}|D_{i_k} \subseteq B)}\right) + \sum_{t=1}^{k-1} \ln\left(\frac{p(y_{i_t}|y_{i_{1:t-1}}, D_{i_{1:t}} \subseteq T_i)}{p(y_{i_t}|D_{i_t} \subseteq B)}\right) =$$

$$= \Delta S^i_{mot}(k) + S^i_{mot}(k-1)$$

Such a formula fits perfectly with online algorithms, i.e.,with algorithms that process the data points frame by frame, like the one proposed in this work.

Given the aforementioned assumptions on the probability distribution of the measurements, it is possible to derive a formula for the increment:

$$\Delta S^i_{mot}(k) = \ln\left(\frac{V}{(2\pi)^{\frac{n}{2}}}\right) - \frac{1}{2}\ln(|\Sigma^i_k|) - \frac{1}{2}d^2 \tag{1.14}$$

with d^2 being the distance computed in (1.13) and $|\Sigma^i_k|$ the determinant of the covariance matrix. In appendix A it is possible to find the computations that are necessary to reach this specific formula.

Every time that a new association is performed, the track's motion score can be updated by using this formula for the increment, which only requires computing the squared Mahalanobis distance between the detection and the track's predicted position.

For what concerns the appearance score, this can be computed as the logarithm of a metric called **Intersection over Union**, which will be described in Section 1.4. This metric will be used considering an appearance feature called "**bounding box**", which consists of a 3D box estimating the volume of the target of the track. Bounding boxes are usually provided in every data set that comprises data used in the context of object tracking.

In this work, the bounding box of the track will be compared with the bounding box of the detection through this metric. The value of the Intersection over Union (IoU) in this case will be an estimate of the probability that the object tracked by the *i-th* hypothesis and the object represented by the detection are the same. Due to the fact that IoU only considers the bounding box of the track and the one of the detection, it is easy to write in a recursive way the appearance score too:

$$S^i_{app}(k) = \Delta S^i_{app}(k) + S^i_{app}(k-1) = \frac{Volume(b_{k-1}) \cap Volume(b_k)}{Volume(b_{k-1}) \cup Volume(b_k)} + S^i_{app}(k-1)$$

where b_{k-1} is the bounding box of the track centered in its position predicted at time k, while b_k is the bounding box of the detection associated with the track at time k. In particular, b_{k-1} is equal to the bounding box of the detection associated with the track in the previous time instant, in order to maintain the recursive nature of the score.

To conclude, the increment of the track's score can be computed in the following way:

$$\Delta S^i(k) = \begin{cases} \ln(1 - P_D) & \text{if } D_{i_t} = \emptyset \\ \omega_{mot}\Delta S^i_{mot}(k) + \omega_{app}\Delta S^i_{app}(k) & \text{otherwise} \end{cases} \tag{1.15}$$

where P_D is the probability that an object will be detected by the algorithm. This probability is decided a priori. As a result, $1 - P_D$ represents the probability that an object has not been detected. This means that in case of missing detection, the increment of the track's score is the logarithm of a probability, which means that whenever a track is not associated with any measurement, its score is diminished of a certain quantity that depends only on the prior probability of detection and not on the type of track or on the position of the track.

1.2.3. Global Hypothesis

A Global Hypothesis is a set of compatible hypotheses coming from the existing trees. The score of a global hypothesis is defined as the sum of the scores of the hypotheses that form the global one. In MHT it is important to compute, at every step, the global hypothesis with the largest score. This global hypothesis will contain the tracks that at time k are more likely to represent the road users involved in the scenario.

Computing the best global hypothesis is an optimization problem that can be formulated as the well-known Maximum-Weight Clique Problem (MWCP). Given a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, a clique is defined as a fully connected sub-graph. Let W be the set of weights of the vertices in V. The MWCP aims at finding the clique with the maximum weight, where the weight of a clique is defined as the sum of the weights of the vertices forming the clique.

In this case, the vertices are the track hypotheses, and their weights are their scores; two vertices are connected by an edge if and only if the two correspondent tracks are compatible, i.e.,if they do not share any observation.

Define the *complement graph* [10] of $G = (V, E)$, indicated as $\bar{G} = (V, \bar{E})$, as the graph whose edge set is given by:

$$\bar{E} = \{(i, j) \ \text{ s.t. } \ i, j \in V,\ i \neq j,\ (i, j) \notin E\}$$

Now the maximum weight clique on graph G corresponds to the maximum independent set on graph $\bar{G}$, that is, the subset of V formed by nodes that are not linked by an edge.

As stated in [10], the MWCP is easier to formulate in this framework:

$$\max \sum_{i=1}^{N} w_i x_i$$

$$\text{s.t.} \qquad x_i + x_j <= 1 \quad \forall (i,j) \in \bar{E}$$

$$x_i \in \{0,1\}, \forall i \in V$$

This formulation is called the *Edge Formulation* and it is the simplest one that can be proposed for the MWCP [10]. It is important to know that the MWCP is an NP-complete problem, which means that in general there are algorithms that return an exact solution in a time that is exponential in the number of vertices in the graph. It is also known that for particular graph structures it is possible to solve the problem in polynomial time, like in the case of *perfect graphs* [10]. In the current framework, the number of vertices might explode, so it is better to look for an exact solution in cases where the number of vertices is small and for an approximate one in cases where the number of vertices is large, like done in [7]. An exact solution is provided by the algorithm described in [15], while an approximated one can be found in [3]. An alternative approach could also be the one of finding the m-best hypotheses, not only the best one [4]. In the current algorithm, the strategy proposed in [15] has been implemented, and the description of this exact algorithm can be found in Chapter 3.

Once a solution for this problem has been obtained, it can be interpreted as the best set of compatible tracks among all the sets of compatible tracks that can be built at time step k.

1.2.4. Pruning

The pruning step is as essential as the computation of the best global hypothesis. A good pruning strategy can speed up the algorithm and make MHT feasible from a time-complexity point of view. Pruning consists of deleting the tracks that are no longer worth to be considered, keeping only the "best" ones. The name comes from the tree structure used in this framework to represent the different track hypotheses: pruning coincides with deleting branches of the tree. To understand which tracks should be pruned, it is necessary to consider the results of the previous step, i.e.,the computation of the global hypothesis. Let k be the current time step, and suppose the best global hypothesis for this time step has been computed.

1. **N-scan Pruning**: for each tree, consider the track in the best global hypothesis

belonging to the current tree and go N steps back in time, that is, consider the positions that the tracks in the tree assume at time step $k - N$, i.e.,their location in the measurement space (x, y, z). Compare those positions with the position of the track in the best global hypothesis at the same time step. Each track whose position differs from the position of the track in the best global hypothesis is pruned. If a tree does not have a track in the best global hypothesis, then it is pruned entirely. In Figure 1.1, an example of 2-scan Pruning is depicted. In the picture, a tree containing 5 tracks (each path from the root to a leaf is a collection of detections, that is a track hypothesis) is depicted. The blue nodes are part of the track in the best global hypothesis. Starting from frame k, i.e.,the last level of the tree, go two steps back in time and prune the tracks that have a different position compared to the track in the best global hypothesis. In this figure the only track that is pruned is the track corresponding to the right branch, which is thus faded.

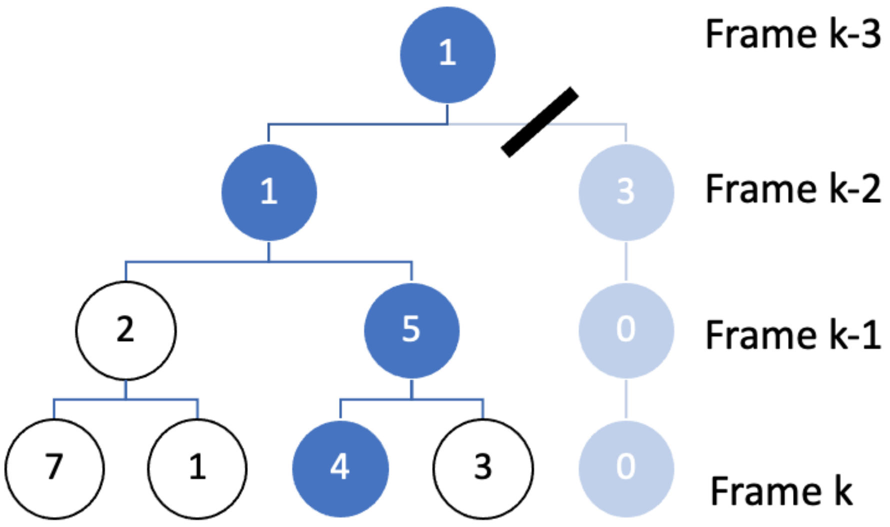


Figure 1.1: Example of 2-scan pruning

The underlying assumption that an N-scan strategy has is that the ambiguities in the detection association step for frames 1 to $k - N$ can be resolved after looking ahead for a window of N frames [5].

2. Prune trees that have grown too large, keeping only a certain number of their tracks, selecting the ones with the highest scores.

The pruning techniques adopted in this work are the same as in [7]

1.3. Motion Models

Kalman Filter assumes complete knowledge of the physical laws governing the modeled phenomenon. In the proposed framework, i.e.,a complex urban scenario, 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. However, in this work the same motion model will be deployed for all the road users, given the fact that an algorithm to distinguish pedestrians from vehicles or cyclists has not been implemented. Two relatively flexible models are those of Constant Speed or Constant Acceleration.

As reported in [13], the main advantage of these simple models is that they are linear, and therefore convergence theorems of Kalman Filters apply. These types of models can perform well whenever the time difference between two frames is not large and the speed of the vehicles is limited, making it possible to approximate the motion of the vehicles with a linear model even though the vehicle might be involved in a turning phase. As a matter of fact, if the time difference between two frames is very small and the scenario is an urban one, the positions of the road users will not change abruptly from one frame to another, in fact they will be very close. In event that either of these two conditions is not satisfied, a different motion model should be considered. The implemented algorithm can also work with different motion models, thus a modification of the motion model might be necessary in different scenarios like the one of a highway.

It is worth mentioning that both the approximations do not include varying accelerations, such as yaw rate. In case this information needs to be included, a Constant Turn Rate and Velocity or Constant Turn Rate and Acceleration motion model could be implemented. However, these motion models will not be considered in the proposed work, but they are analyzed and compared thoroughly in [13].

The Constant Speed model is as follows:

$$x(t) = \begin{bmatrix} x(t) \\ v_x(t) \\ y(t) \\ v_y(t) \\ z(t) \\ v_z(t) \end{bmatrix} \qquad F = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The Constant acceleration model is as follows:

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

The convergence of the kalman filter is guaranteed by the fact that the system is fully controllable from the input sensor, fully observable and the correlation matrix between the input and process noises is set to zero. These are three sufficient conditions that guarantee that the Difference Riccati Equation showed in 1.1 converges to a solution and that the corresponding asymptotic gain of the kalman filter is such that the filter is asymptotically stable.

1.4. Intersection over Union

Viscando's stereovision sensors, and several other sensor technologies, measure the extent of the objects in three dimensions, yielding a 3D bounding box. One critical advantage of having bounding boxes for each detection is the possibility of incorporating information about the dimension of an object and the 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:

$$IoU(b_1, b_2) = \frac{Volume(b_1) \cap Volume(b_2)}{Volume(b_1) \cup Volume(b_2)} \tag{1.16}$$

This metric will be used in the computation of the score of the track, as explained in Section 1.2.2. Given its formula, it is clear that IoU becomes useful if the available bounding boxes are sufficiently accurate. This means that a better bounding box approximation should yield better tracking.

1.5. Bootstrap Confidence Intervals

Bootstrapping is a "resampling" technique that allows to estimate the distribution of a statistic, that is, an estimator of an unknown parameter. The name "resampling" techniques refers to algorithms that resample with replacement from a random sample in order to extract a specific type of information.

Let θ be an unknown parameter, and let $\hat{\theta}$ be the statistic used to estimate the parameter. θ can be seen as a parameter of a population of independent and identically distributed random variables that follow a probability distribution which in most of the cases is unknown:

$$\vec{X} = (X_1, X_2, ..., X_n) \overset{iid}{\sim} F$$

where F is the unknown cumulative distribution function of each component X_i of the vector. As a consequence, θ depends on the unknown distribution represented by F and $\hat{\theta}$ too, since $\hat{\theta}$ is a function of the random vector $(X_1, X_2, ..., X_n)$.

A confidence interval for θ can be built by means of the following property:

$$\mathbb{P}[\hat{\theta} - (\hat{\theta}_{\alpha/2} - \theta) < \theta < \hat{\theta} - (\hat{\theta}_{1-\alpha/2} - \theta)] = 1 - \alpha \tag{1.17}$$

Here the convention used is that the index of the quantile indicates the area under the distribution curve after the quantile. The main issue with this formula is that it cannot be used because the distribution of the estimator $\hat{\theta}$ is unknown and so are its quantiles.

Supposing to have a sample $\vec{x} = (x_1, x_2, ..., x_n)$, it is known, thanks to the **Glivenko-Cantelli Theorem**, that the empirical cumulative distribution $\hat{F}$ of this sample converges asymptotically to F as n increases:

$$\hat{F} \xrightarrow{a.s.} F$$

In addition to this, resampling with replacement from the sample $\vec{x} = (x_1, x_2, ..., x_n)$ allows for the generation of a new sample $\vec{x}^* = (x_1^*, x_2^*, ..., x_n^*)$. Let $\hat{\theta}^*$ be the estimator $\hat{\theta}$ computed over this specific random sample. The sample $\vec{x}^*$ has been "bootstrapped" from $\vec{x}$, thus $\vec{x}^*$ is called "*bootstrap sample*" and $\hat{\theta}^*$ is called *bootstrap estimator*. It is clear that $\hat{\theta}^*$ is a random quantity that depends on the *bootstrap sample* that is resampled from $\vec{x}$. Since the numerousity of the sample is n, the number of different *bootstrap samples* that can be resampled is n^n. As a consequence, the *bootstrap sample* is uniformly distributed over the n^n possible sequences that can be obtained from resampling with replacement.

Supposing to draw all the possible *bootstrap samples* then it would be theoretically possible to compute all the values of $\hat{\theta}^*$, from which the empirical distribution of $\hat{\theta}^*$ can be derived. The **Glivenko-Cantelli Theorem** guarantees that the empirical distribution of $\hat{\theta}^*$ converges asymptotically to the distribution of the estimator $\hat{\theta}$. Thus, the quantiles of the distribution of $\hat{\theta}$, necessary for the computation of the confidence interval, can be estimated through the quantiles of the distribution of $\hat{\theta}^*$, by exploiting the reasoning above and the **Glivenko-Cantelli Theorem**.

The framework aforedescribed is called the **Plug-in Principle**, which indeed states that it is possible to estimate a feature of an unknown distribution (e.g., mean, quantiles, ecc) by estimating the same feature (e.g., mean, quantiles, ecc) on the empirical distribution. The name **Plug-in Principle** comes from the fact that this principle allows to substitute the "real world" in which θ is the unknown parameter and $\hat{\theta}$ the estimator, with the "bootstrap world", in which $\hat{\theta}$ is the target and $\hat{\theta}^*$ is the estimator.

However, it is in practice unfeasible to draw all the n^n random samples, thus the bootstrap empirical distribution cannot be obtained in an exact way but it needs to be estimated via **Monte-Carlo sampling**. Fixing a number B of random *bootstrap samples* drawn by resampling with replacement from $\vec{x}$, thanks to the **Law of Large Numbers** it is possible to state that:

$$\mathbb{P}_{\hat{F}}(\hat{\theta}^* \leq t) \approx \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{(\hat{\theta}^*(\vec{x}_b^*) \leq t)}$$

$$\text{with} \quad \vec{x^*}_b \overset{iid}{\sim} \hat{F} \quad b = 1, ...B$$

with $\hat{F}$ being the empirical cumulative distribution function of the components of the vector $\vec{x^*}_b$.

To sum up, the quantiles of the distribution of $\hat{\theta}$ can be estimated through the quantiles of the distribution of $\hat{\theta}^*$ thanks to the Plug-in Principle and the Glivenko Cantelli Theorem. Given the extremely high computational load that would be faced to compute the distribution of $\hat{\theta}^*$, it is possible to estimate this distribution by coupling a bootstrap resampling strategy with Monte-Carlo sampling, thus exploiting the Law of Large Numbers.

Once the empirical distribution of $\hat{\theta}^*$ has been estimated, the next step is to compute the quantiles of this distribution and put them in place of the quantiles of the estimator as the Plug-in Principle allows to do.

The confidence interval can now be rewritten replacing the unknown quantities in (1.17) in the following way:

$$(\hat{\theta}_{\alpha/2} - \theta) \leftrightarrow (\hat{\theta}^*_{\alpha/2} - \hat{\theta}) \qquad (\hat{\theta}_{1-\alpha/2} - \theta) \leftrightarrow (\hat{\theta}^*_{1-\alpha/2} - \hat{\theta})$$

where $\hat{\theta}^*_\alpha$ is the quantile of order α of the estimated empirical distribution.

To conclude, the bootstrap confidence interval obtained for the unknown parameter θ becomes:

$$CI_\alpha(\theta) = [2\hat{\theta} - \hat{\theta}^*_{\alpha/2}, 2\hat{\theta} - \hat{\theta}^*_{1-\alpha/2}] \tag{1.18}$$

with $\hat{\theta}$ being the estimator computed on the original sample $\vec{x}$.

2 | Description of the Data Set

This chapter introduces the data set that will be used to validate the Multiple Hypothesis Tracker that will be implemented. In addition to this, there will be a short discussion about common issues that emerge in every tracking problem.

The provided data is a collection of information from 1 stereovision sensor, 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. The 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
  - length of the bounding box
  - width of the bounding box
  - 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 processed in real-time in the sensor's embedded computational unit and removed immediately thereafter. This ensures the GDPR compliance of Viscando's traffic measurement solutions. In this 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 the this work, but can rather be a term of comparison.

The way the data are collected strongly affects the result of any type of tracking algorithm. The well-known problems that are underlined in the literature can be found in this data set too. It is important to describe and anticipate them because they represent the main challenges to the multi-object tracking problem:

1. **Fused Detections:** in crowded scenarios, it happens that two pedestrians or other road users and stationary objects are indistinguishable for the sensor and thus they are detected as a unique object. An example of this defection is shown below in Figure 2.1.

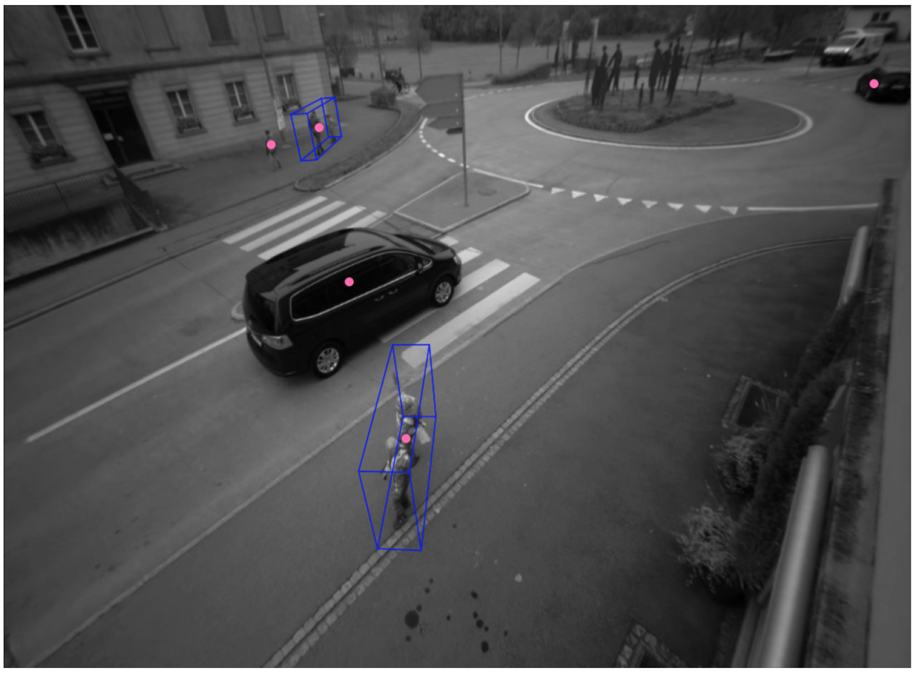


Figure 2.1: Frame number 10. The blue bounding boxes represent objects that have been merged.

This problem can cause tracks to be interrupted or it can result in a track that shifts its target. Another example is also portrayed in Figure 2.3. This type of problem has already been addressed in [11]. In this work a new strategy to identify the cases in which this issue will appear and to maintain the identity of objects targeted by the tracks will be proposed.

2. **Occlusion of Objects**: a common problem in tracking applications is how to track a road user that becomes partially or completely occluded [6]. The main reason this happens is the presence of obstacles. As in every urban scenario, also in this case there are some obstacles that are partially responsible for occlusion but also for some of the problems mentioned in this section. This is a complication that can be solved as long as the occlusion does not last for too many frames: nearly all tracking algorithms terminate a track with loss of identity after a long period of occlusion [11]. In particular, in this context the main obstacles are road signs and statues in the middle of the roundabout, as Figure 2.2 shows.

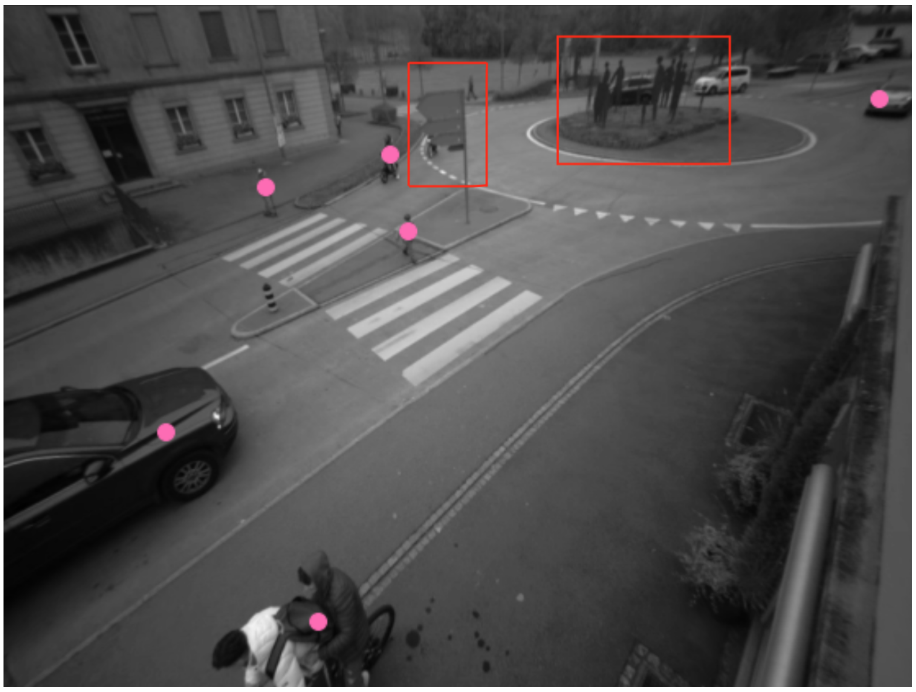


Figure 2.2: The road sign and the statues in the middle of the roundabout are occluding two vehicles.

3. **Missing Detections:** due to different reasons, like distance from the sensor or inaccuracies in the sensor, an object might not be detected even if it is not occluded. This is a problem at the basis of every tracking system, because it is very difficult to determine the next position of the track without any data associated with it, unless the Kalman Filter has converged. Even in this case, the position predicted by the filter becomes too noisy and uncertain if the detection is missed for multiple frames. An example of a missing detection that is not caused by the presence of obstacles but by sensor accuracy is depicted in Figure 2.3.

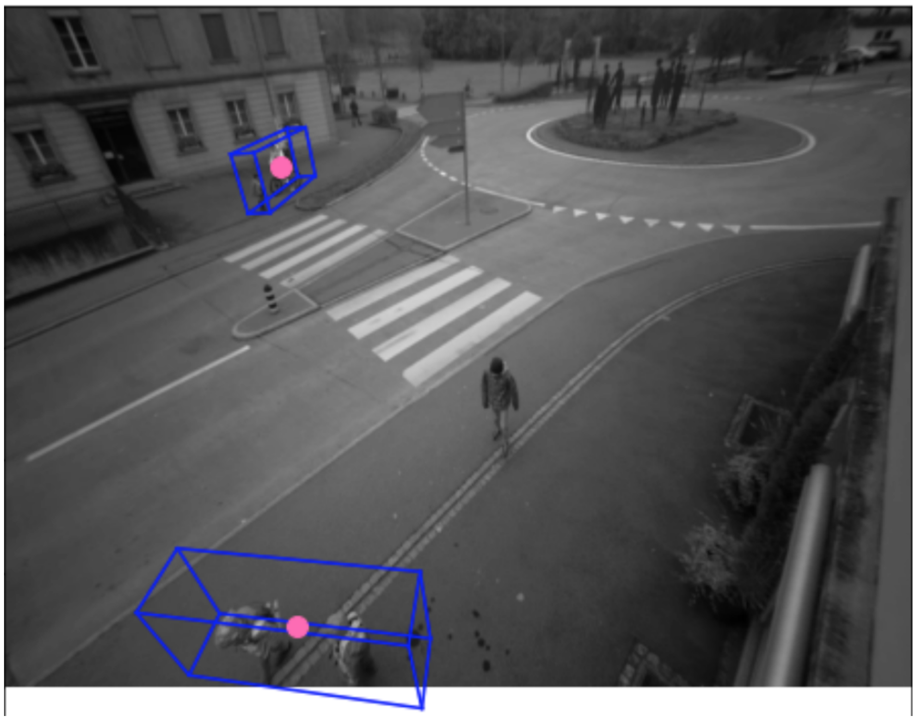


Figure 2.3: Frame number 131. One pedestrian is not detected, the others are merged in a bounding box given their proximity.

4. **False Detections:** no sensor has an accuracy of 100%, meaning that sometimes some detections do not represent real targets. These type of data usually generate tracks that are called "*False Alarms*". False Alarm tracks are easy to spot and eliminate if the noisy detections only exist for few frames, but they can create an issue if they are present in consecutive frames. For example, it can happen that a valid track of a real road user can jump to a nearby false detection and, as a result, lose the real object. Figure 2.4 below shows an example of a detection coming from a stationary object and not from a road user.

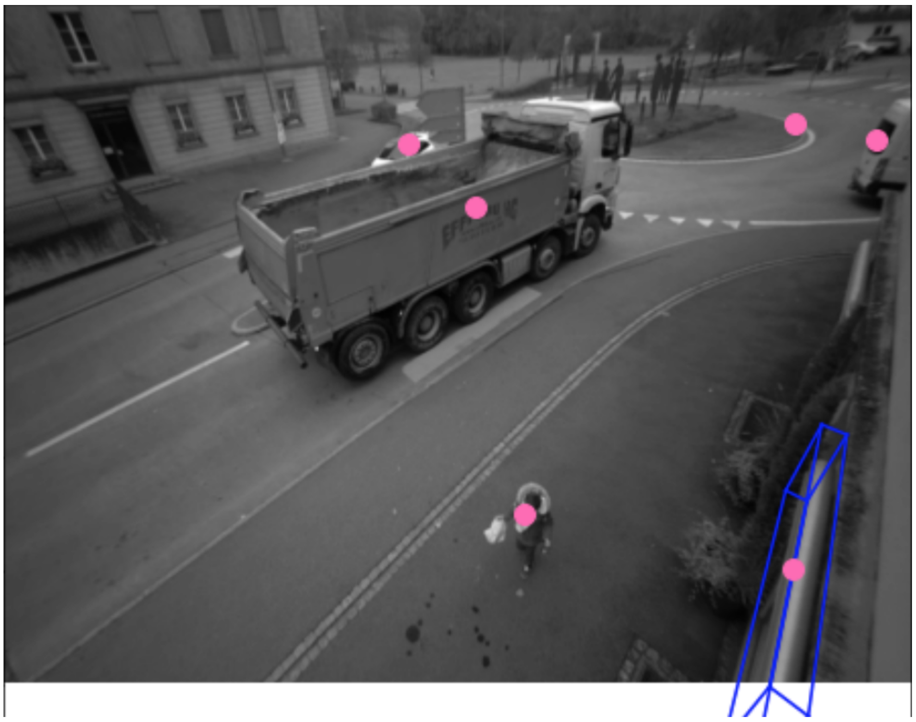


Figure 2.4: Frame number 550. The blue bounding box shows a detection related to an object which is not a road user.

As stated in the introduction, the assumption of this work is that the proposed algorithm will be able to address these problems and solve them, outperforming the conventional single hypothesis tracker in cases where the problems listed above make it hard to perform correct associations.

3 | Implementation of the Algorithm

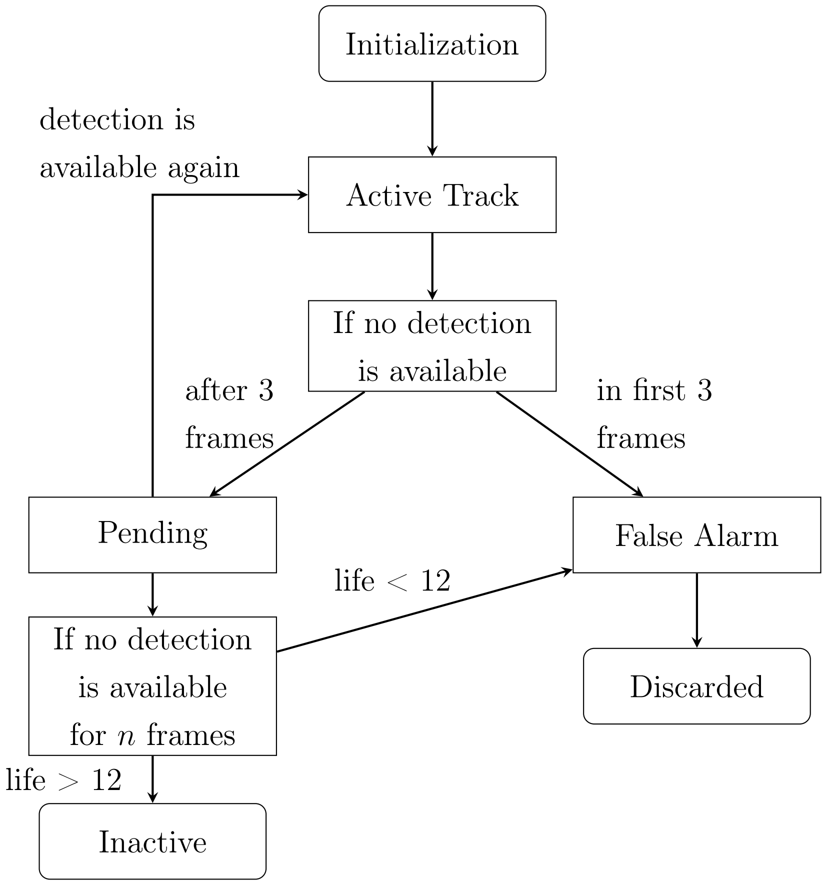
This chapter is thought of as an explanation of how the algorithm to solve the tracking problem has been implemented using Multiple Hypothesis Tracking (MHT), along with a strategy for detecting fused objects. In particular, this strategy will use bootstrap confidence intervals, which have been introduced and explained in Section 1.5. The following section will show in detail how Multiple Hypothesis Tracking works.

3.1. Implementation of Multiple Hypothesis Tracking

As anticipated above, this section will go through all the steps that MHT consists of. The fundamental elements of MHT are the following three entities:

- **Road User:** it represents the information coming from one detection as they are described in Chapter 2. A road user can correspond to a pedestrian, a cyclist, or a vehicle detected by the sensor.
- **Track:** This entity embodies the main characteristic of one track hypothesis. As such, it will be characterized by the list of **positions** of the track and the list of **detections** associated with the track, as well as the **Kalman Filter** for the track. It is labeled with a unique **ID** and it also includes information on the tree it belongs to. Moreover, a track has a **status** that describes its behavior: Active, Pending, Inactive or False Alarm. Active tracks should represent targets that are in the roundabout, while inactive tracks' targets left the roundabout in previous frames. Pending tracks refer to targets that might be occluded or for which the detection might not be available, while False Alarms are those tracks that do not represent road users.

  Whenever a track is initialized, its status is set to *active*. If the track is not associated with any detection in one of the first three frames of activity, the track's status is set to *False Alarm*. After these three frames, whenever the track is not associated with any detection, its status is set to *Pending*. A track can be pending for a maximum of five frames, or twelve if the track has been active for a long time. After the track has been pending for the maximum allowed number of frames, its status is either changed to *Inactive*, if its lifetime is longer than twelve frames, or *False Alarm* if its lifetime is shorter than twelve frames. The flow chart below shows how the tracks can move from one status to another.



The choice of these parameters is arbitrary. In every tracking implementation the strategies can be different, and they also depend on the scenario and on the precision of the sensors. For instance, in [7], a track is considered *inactive* after fifteen frames, but the framework considered is an only-pedestrian one. In [1], S. Blackman states that track deletion is determined after a number of frames which can be typically 4 or 7 (it is 5 in this implementation, 10 in special cases), and track confirmation is also determined by rules such as three detections in four frames.

- **Tree** This entity serves as the **tree structure** discussed in Chapter 1. It stores all the **tracks** that belong to the same tree. Every tree is distinguished by a unique **ID**.

A tree has a **status** which depends on the status of its tracks: Active or Inactive. A tree is **Inactive** if all its tracks are inactive, otherwise it is considered **Active**.

Now that the entities have been listed and explained, it is possible to enter into details and analyze how the algorithm actually works. The algorithm iterates over each frame and extracts from the data set the rows corresponding to the detections belonging to that frame. Chapter 2 describes what type of data every detection consists of.

Let k be the index of a current frame, the following are the operations that the algorithm performs on such a frame:

- Iterate over the active and pending tracks. For each track, the subsequent steps are made:

  1. **Associations:** Associate detections of frame k with tracks using the criteria described in Section 1.2.1, i.e.,a track is associated with all the detections that fall in its gating area. More precisely, if the track has been created more than 12 frames ago, predict the next position of the track through Kalman Filter and use this position as the center of the gating area. However, if the track was initialized within the past 12 frames, use the detection associated with the track at frame $k-1$ as the center of the gating area without doing any prediction because the filter still needs to start converging before being used. This is not controversial since the frames differ by only 0.08 s, which implies that the difference between the next position and the last position is very small given the fact that the current scenario is an urban one. If other scenarios need to be considered, for example a highway, it would be appropriate to increase the size of the gating area whenever its center is not the predicted position but the last detection associated with the track.

     It is also important to mention that, since the Intersection over Union between the track and the detection needs to be computed and the logarithm of this quantity is a factor in the score of the track (Section 1.2.2), those associations for which the Intersection over Union is zero have to be discarded. This is not a typical condition in Multiple Hypothesis Tracking, since it is only due to the type of appearance score used in this work. However, as mentioned above, this constraint may be too tight in a non-urban scenario like a highway. In such a scenario, it is fundamental to change the way Intersercion over Union is computed: it is indeed appropriate to center the last associated bounding box in the position predicted by the tracker at frame k and compare it with the bounding box of the detection associated with the track at frame k. It might be necessary in this case to avoid computing this metric in the first frames of the track, that is, when the filter of the track is not reliable in its predictions.

2. **Score**: For each association made in the previous step, compute the quantity $\Delta S^i(k)$ (where i is the index of the current track) as described thoroughly in section 1.2.2.

3. **Update**: Iterate over the detections associated with the track. For each association except the last one, create a copy of the current track and update the filter of this copy with the new detection. In this way a new track hypothesis has been created. This new track is stored in the same tree of the current track, and it is declared incompatible with all the tracks in the tree and vice versa. Moreover, if the detection has been previously associated with other tracks, declare the new track incompatible with those ones and vice versa. The current track is updated with the last association performed.

   If the track has not been associated with any detection, set its status to *Pending*. If the track reached the maximum number of frames allowed for being Pending, then set its status to *Inactive* or *False Alarm*, based on the lifetime of the track, as explained above. If the track is Inactive and is incompatible with no track, then the track can be considered a real target's trajectory.

- Now all the tracks have been updated and their score is an index of how likely they are to represent a real target. As a consequence, it is possible to compute the best global hypothesis, solving the optimization problem described in Section 1.2.3. A slightly modified version of the algorithm proposed in [15] is adopted to solve this problem. The tracks considered for this step are the active ones and the inactive ones that are still *incompatibles* with other tracks. Let N_k be the number of tracks considered. At first, sort in descending order the tracks based on their score. Then, iterate from the track with the lowest score to the one with the highest score. Let i be the index of the iteration to be analyzed. Consider the subset containing the tracks from the *i-th* from the bottom to the last track:

$$(v_{N_k-i}, v_{N_k-(i-1)}, ..., v_{N_k})$$

the tracks are indicated with v_j because in this optimization problem they represent the vertices of a graph, as explained in Section 1.2.3. The goal is to obtain the clique with the highest score containing v_{N_k-i} (the score of a group of tracks is the sum of the scores of its tracks, Section 1.2.3) on the subgraph induced by $(v_{N_k-i}, v_{N_k-(i-1)}, ..., v_{N_k})$. Define a *working set* W_0 made of the elements of the subset $(v_{N_k-i}, v_{N_k-(i-1)}, ..., v_{N_k})$ that are compatible with v_{N_k-i}.

$$W_0 = (v_{N_k-i}, v_{N_k-(i-1)}, ..., v_{N_k}) \cap \text{tracks compatible with } v_{N_k-i}$$

In the previous iterations the best cliques for all the elements in W_0 have already been found . This means that either one of this cliques is made of tracks compatible to v_{N_k-i}, or a new clique needs to be found for the v_{N_k-i}. In both cases, it is necessary to iterate on every element of W_0. Let j be the first element of W_0 and consider its best clique, labeled as C_j. If $C_j \subseteq W_0$, then $C_j \cup \{N_k - i\}$ is the temporary best clique for v_{N_k-i}. Otherwise, put the current element in a temporary clique and build another working set W_1 given by the intersection of the current working set W_0 and the elements compatible with *j-th* track. Repeat the same operation, iterating over the new working set W_1. Eventually a clique for *i-th* track will be found, but it is not guaranteed to be the best. This is why the same steps described above have to be repeated starting from the second element of the working set W_0. At the end, only the best clique will be kept, since this is the desired output.

Once the clique for the last track, i.e.,the one with highest score, is found, then the global hypothesis coincides with the clique with the highest score among the ones that have been found.

In addition to the tracks in this clique, the inactive tracks that are incompatible with no track enter the global hypothesis as well.

- After computing the best global hypothesis, it is possible to prune the trees using the techniques described in Section 1.2.4. More precisely, iterate over all the Active trees. If there is a track of the current tree in the best global hypothesis, prune the tree using **N-scan Pruning**, otherwise prune the entire tree. In addition to this, whenever a tree is too large, prune it by keeping only its N_{best} tracks, i.e.,the first N_{best} tracks in terms of scores, after ordering them from the one with the largest score to the one with the smallest.

When all the frames have been processed, the tracks in the best global hypothesis should represent the trajectories of the real targets that have been in the roundabout. It is important to underline that whenever a track becomes *inactive* it means that its target has left the roundabout. As such, the track can be directly added to the global hypothesis if it is compatible with all the existing tracks, otherwise it has to be involved in the computation of the best clique.

The pseudo-algorithm below can help in better understanding how the algorithm works:

Algorithm 3.1 **Multiple Hypothesis Tracking**

```
1: for $k$ in 0,...,749 do
2:   Extract Detections from frame $k$
3:   for $track_i$ in active and pending tracks do
4:     Associate with $track_i$ the detections that fall in its gating area
5:     for $det_j$ in associatied detections do
6:       Compute $\Delta S^i(k)$ following instructions in Sec. 1.2.2
7:       Update a copy of $track_i$ with $det_j$
8:     end for
9:     if No detections are associated with $track_i$ then
10:      Set the status of $track_i$ to Pending
11:    end if
12:    if $track_i$ has been pending for more than 5 frames then
13:      if lifetime of $track_i$ is less than 12 frames then
14:        Set status to False Alarm
15:      else
16:        Set status to Inactive
17:      end if
18:    end if
19:  end for
20:  Solve Optimization problem to find best global hypothesis
21:  for $tree_n$ in active trees do
22:    if no tracks of $tree_n$ is in Best Global Hypothesis then
23:      Prune all the tree
24:    else
25:      let $\overline{track}_n$ be the track in the best global hypothesis
26:      prune tracks in $tree_n$ that differ from $\overline{track}_n$ at frame $k - N$ (N-scan Pruning)
27:    end if
28:    if if tree is too large then
29:      prune tree keeping only the best $N_{best}$ tracks in terms of score
30:    end if
31:  end for
32: end for
33: The tracks in the best global hypothesis together with the inactive tracks that are
    compatible with all existing tracks represent the targets in the roundabout
```

This is how MHT works in the implemented algorithm. It is important to underline that this version of Multiple Hypothesis Tracking is not able to deal with fused detections, meaning that if in a frame there are two or more road users that are detected as one, a problem that has been emphasized in Chapter 2, the tracks corresponding to the "fused targets" will be likely to become incompatible, because in the event that the fused detection fell into their gating area, they would both be associated with it, causing the two tracks to become incompatible. This issue might have been taken into account by [7], thanks to a property that has not been implemented in this algorithm: in addition to the hypotheses generated as a result of the association phase, each track should generate another hypothesis, corresponding to the case of *missing detection*. As a result, in the case described above, the two tracks that are associated with the same detection (representing two merged road users) should also have the possibility of not being associated with any observation. Three observations need to be made regarding this strategy:

1. If the fused detection is present for multiple frames, as is constantly the case in the data set analyzed in this work, there will be too many tracks that have been created and that will compete for the same detections. In particular, the number of tracks involved would increase exponentially: suppose that at frame k there are two tracks that are associated with the same detection (representing fused objects), then at the next frame there will be four tracks that would be very close to each other and that would probably be associated with the same detection at frame $k + 1$; this means that at the next frame there will be eight tracks competing for the same detection. It is thus clear that the computational complexity of the algorithm would become prohibitive even if a good pruning strategy is adopted.

2. In a scenario that is not densely populated, generating a new hypothesis with a missing detection is not always necessary. There are cases in which road users are sufficiently distant from any other type of road user; thus, there is no need for an hypothesis of missing detection if the detection is actually available.

3. This strategy does not face the problem directly but it is a conservative strategy that is generally adopted by the algorithm. It is thus necessary to develop a process that tackles this specific issue, given its frequency in every urban scenario and its gravity.

For these reasons, the proposed algorithm does not adopt this strategy but instead uses another reasoning that exploits the information contained in the bounding boxes related to the detections. In particular, the volumes of these bounding boxes are used to detect if two objects have been merged into one detection, as the next section will thoroughly explain.

It is worth mentioning that in very cluttered scenarios, like an only-pedestrian one, adding one hypothesis corresponding to the case of missing detection can be useful, as [7] shows, if coupled with a more aggressive pruning strategy.

3.2. A Strategy for Detecting Merged Objects

The aforementioned strategy implemented to detect cases of objects fused in the same bounding box and to tackle this issue will be introduced and explained in this section. As previously stated, a track hypothesis is a collection of positions that are derived from the detections associated with it. Each detection also has information on the bounding box of the object: the width, the length and the height. This information is a measurement of the object's volume. This means that each track also contains a sample of volumes of its target, given by all the estimates of the object's volume coming from previous detections. The idea of the proposed strategy is to estimate a 95% confidence interval for the volume of the target starting from this sample. The distribution of the data contained in the sample is not known, and neither are the quantiles of the distribution. To overcome this problem, it is possible to build the so-called **Bootstrap Confidence Interval**, a topic that has been introduced and explained in Section 1.5. To quickly recall, the quantiles of the unknown distribution are substituted by the quantiles of the distribution of the estimator of the volume (**Plug-in Principle**), which in this case will be the sample mean. This distribution is in turn estimated through its empirical distribution, obtained by resampling with replacement from the given data set and computing each time the value of the estimator (**Bootstrapping**). The collection of the values of the estimator gives the empirical distribution, from which the quantiles are obtained.

Once the confidence intervals have been computed, they can be used for the purpose of detecting merged objects. In particular, whenever two tracks are associated with the same detection, a confidence interval for their volume will be computed. If the volume of the bounding box of the associated detection is inside at least one of the two confidence intervals, then the object is not considered fused. If the volume of the bounding box of the associated detection is bigger than the upper bound of both intervals, then the two underlying objects are considered fused. Indeed, it is possible to see from Figure 3.1 that whenever two objects are merged into one detection, the volume of the resulting bounding box is way bigger than the volume of either the previous boxes.

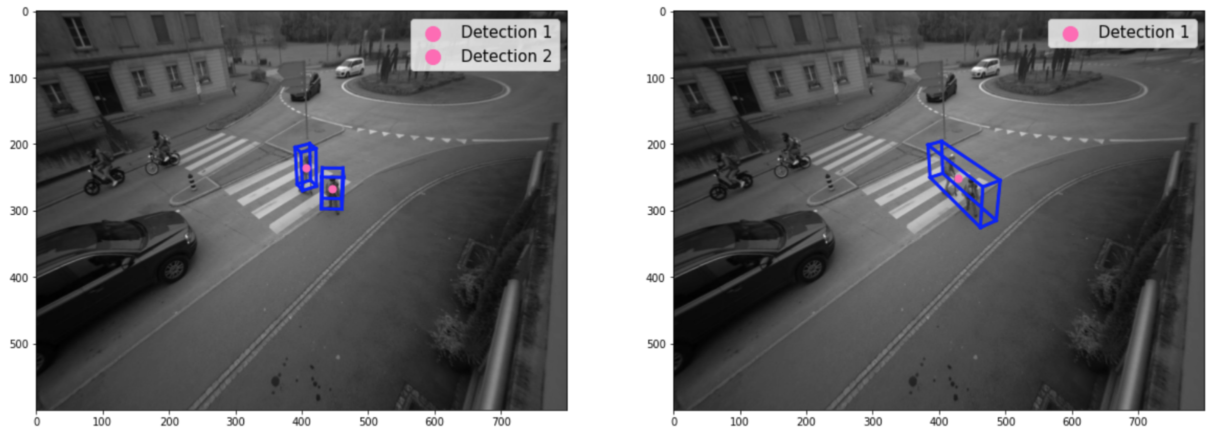


Figure 3.1: Image on the left shows two separated road users. In the image on the right, corresponding to the subsequent frame, the sensor fails to separate the road users, which leads to a detection with a larger bounding box.

Given the fact that for each track a 95% confidence interval is computed, and given the fact that two confidence intervals are considered together, the significance level reached with this strategy is 90%. To achieve a 95% confidence level on the whole strategy, a *Bonferroni correction* should be implemented and two confidence intervals with a 97.5% level of significance should be computed.

If two objects are considered fused by the aforementioned scheme, then the algorithm predicts the position of the two tracks representing the fused targets, and if the predicted position is inside the fused bounding box, then this position is used as the next one for the track and the Kalman Filter does not receive any data (the filtering step is skipped). On the other hand, if the predicted position is outside the bounding box of the detection, then the Kalman Filter of the track is updated with a weighted mean between the predicted position and the position given by the detection:

$$\text{New Position} = 0.7 \cdot \text{Predicted position} + 0.3 \cdot \text{Detection}$$

The reasoning behind these weights is that the position coming from the fused detection is not reliable because it corresponds to the center of a bounding box containing two road users. This often implies that the position of the fused detection is in-between the fused targets. As a result, it could be correct to give more weight to the predicted position than to the detection. On the other hand, the predicted position itself could be wrong, especially after a few frames in which the road users are merged, because in these cases the variance of the filter tends to increase quickly. A weight of 0.7 for the predicted position and 0.3 for the detection seemed a priori to be a good trade off to reduce the errors that might come from the detection and the prediction.

It is worth mentioning that if two tracks are considered fused for a certain number of frames, in the subsequent frame, when they are no longer fused, the Intersection over Union constraint is not applied to any association. In this way, hopefully, the tracks will have the opportunity to reconnect with their target. As a matter of fact, if there is a situation of fused detections that goes on for more than few frames, it is very likely that the track will be significantly distant from the detections of its target. This is due to the fact that the road users change direction or speed, or both, during the fusion period, but none of these changes are detected if the road users are merged. As a result, the Intersection over Union between the bounding box of the track and the one of the target might be equal to zero once the "fusion period" is finished.

To conclude, the proposed algorithm couples Multiple Hypothesis Tracking with a strategy to detect fused objects that is based on bootstrap confidence intervals for the volume of the objects. The next chapter will show the results that this algorithm obtains when tested on the data set described in Chapter 2.

4 | Results

In this chapter the performances of the proposed algorithm will be analyzed. What follows first is a brief introduction to get familiar with how the results will be presented.

4.1. Introduction to the Results

Before showing the results, it is more appropriate to specify some properties of the algorithm and explain the type of plots that will be used to discuss and analyze its performance. First of all, the results are referred to the algorithm that uses a **constant speed model** as motion model for the Kalman Filter. Later on there will be a comparison between constant speed and constant acceleration as motion models (Section 5.2). Moreover, the results shown in this section have been obtained with the parameters reported in Table 4.1. It is worth recalling their meaning:

- $\boldsymbol{d_{th}}$: threshold for the squared Mahalanobis distance between a detection and the position of the track, Section 1.2.1. It establishes the size of the gating area of the track. The value is chosen according to [7];
- $\boldsymbol{P_D}$: Probability of detection. It represents the probability that an object that appears in the scenario will be detected by the sensor. It is involved in the computation of the score of the tracks, Chapter 1.2.2. Its value is chosen a priori according to [7];
- **V**: image area, estimated through available data on dimensions of the images. It is as well involved in the computation of the score of the tracks, Section 1.2.2;
- **N-scan**: This parameter refers to the pruning strategy described in Section 1.2.4. It establishes how many frames back in time the algorithm has to go to check if there is a difference between the track in the best global hypothesis and another track in the same tree, in order to decide if the other track can be kept or discarded. Its value is chosen according to [7];
- $\boldsymbol{N_{best}}$: Number of best tracks to keep in case a tree is too large and needs to be

pruned, Section 1.2.4. Its value is chosen according to [7];

Parameters MHT				
d_{th}	P_D	V	N-scan	N_{best}
6	0.9	480000	5	100

Table 4.1: Values for parameters of MHT

Now that the parameters have been fixed, it is possible to explain how the tracks will be plotted in the following sections. As discussed previously, a track is a collection of positions given by the Kalman Filter once the data has been fed to it. Moreover, together with each position, the track also has a bounding box, which comes from the data. Given this information, a possible way of representing tracks is by plotting all the positions separately and subsequently, together with the correspondent bounding box, which will be centered in the position of the track in that frame. Figure 4.1 is an example of a track collecting the positions of a pedestrian approaching the crossroad.

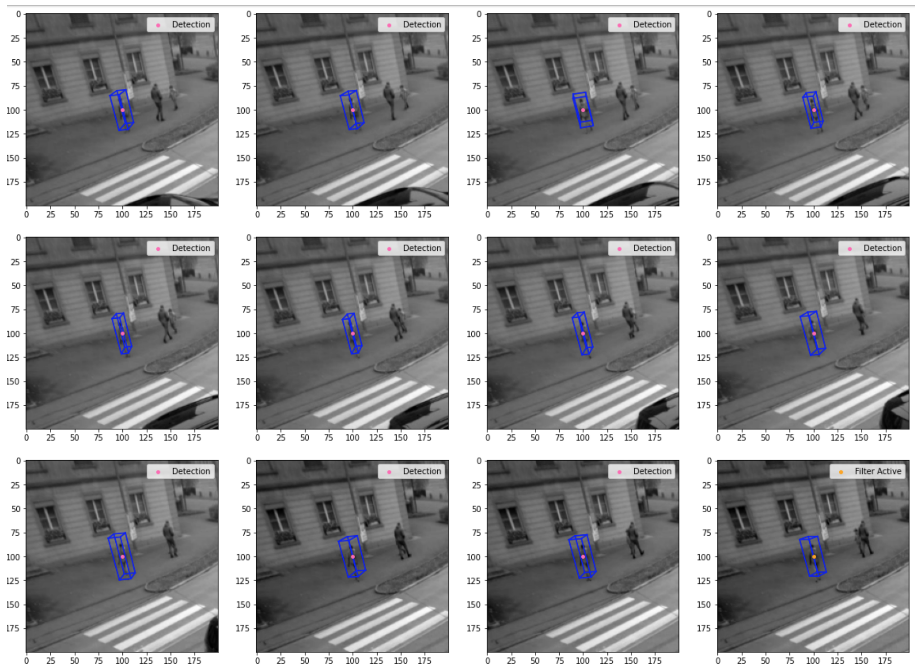


Figure 4.1: Example of track visualization. Each subplot corresponds to a certain frame. The represented frames are subsequent.

Every track that will be represented is a track in the best global hypothesis. Every image will also portray a legend that refers to the status of the track during that specific frame. In particular, the labels in the legend can be of four types:

- **Detection**: This is the label for the first positions of the track. During its first frames the track does not use the position given by the Kalman Filter because the filter is not reliable yet. As a consequence, these positions are exactly equal to the measurements coming from the detections. The color of the point will be pink;
- **Filter Active**: This label stands for a position that is the result of the filtering process described in Section 1.1. The track has been associated with a detection, that has been given to the filter which outputs its estimate of the state variable. This estimate is the position of the track in that frame, which will be represented as an orange dot;
- **Filter Pending**: In this case, the track is pending because it has not been associated with any detection. Thus, its positions are the ones predicted by the filter. The colour of the dot will be green;
- **Filter Fused**: It refers to a detection that the algorithm recognizes as two merged objects, using the strategy discussed in Section 3.2. The colour of the dot will be blue.

4.1.1. How to Analyze the Performance of the Tracker

A key topic for every tracking system is how to analyze the results once the system has been applied to a data set. A very common approach is to use **ground truth trajectories**, that is, a set of trajectories that have to be considered true and should be compared to the results to understand if the performance of the tracking system is sufficiently good. In this work, ground truth trajectories are not available, so it is not possible to use the general framework to assess the quality of the proposed algorithm. The construction of ground truth trajectories is not an easy task, but more importantly, annotating the targets by looking at the pictures is not a reliable method, and it can present some cases of ambiguities, as shown in [8]. Moreover, even if ground truth trajectories were available, the lack of a standardized benchmark would make it hard to directly compare the different tracking systems [9]. Last but not least, in real-time field testing the ground truth trajectories are not available [14], thus developing an alternative strategy is a good idea. For all the reasons mentioned, in this work ground truth trajectories will not be built and thus will not be used to evaluate the proposed tracking system.

Since ground truth trajectories will not be available, the performance of the algorithm will be analyzed by showing how it deals with problematic cases like the presence of fused or occluded objects. In addition to this, even though ground truth trajectories are not available, it is still possible to count how many tracks of the following two categories the tracker produces: **False Positive** and **Identity Switches**. According to [9], a *False Positive* track is a track that does not match any ground truth trajectory, while an *Identity Switch* occurs when a track shifts its target. As already stated, ground truth trajectories are not available, but it is still possible to understand if a track falls into one of those two categories by simply looking at it and checking which object is targeted by the track. If the object is not a road user, then the track is a *False Positive* one. If the track changes between two or more valid targets, then an *Identity Switch* has occurred.

That said, in the following sections the algorithm will be analyzed by looking at the tracks given as output, detecting weak spots and common pitfalls as well as by assessing its strengths.

4.2. Analysis of the Results

In this section the results obtained by the proposed algorithm will be discussed using the methodology and the type of plots explained in the previous section. This section will begin with visualizing the tracks that form the output of the tracking system from above, reported in Figure 4.2.

The figure depicts all the tracks that are considered part of the best global hypothesis after all the frames from the data set have been processed. The blue dot represents the starting point of each track and the red the end point, as stated by the legend.

It is important to recall that the sensor from which the data are collected is in the bottom right part of the roundabout, and that the traffic sign in the middle of the scene and the statues on the roundabout constitute obstacles that cause objects to be occluded, as discussed in Chapter 2 and as shown in Figure 4.3.

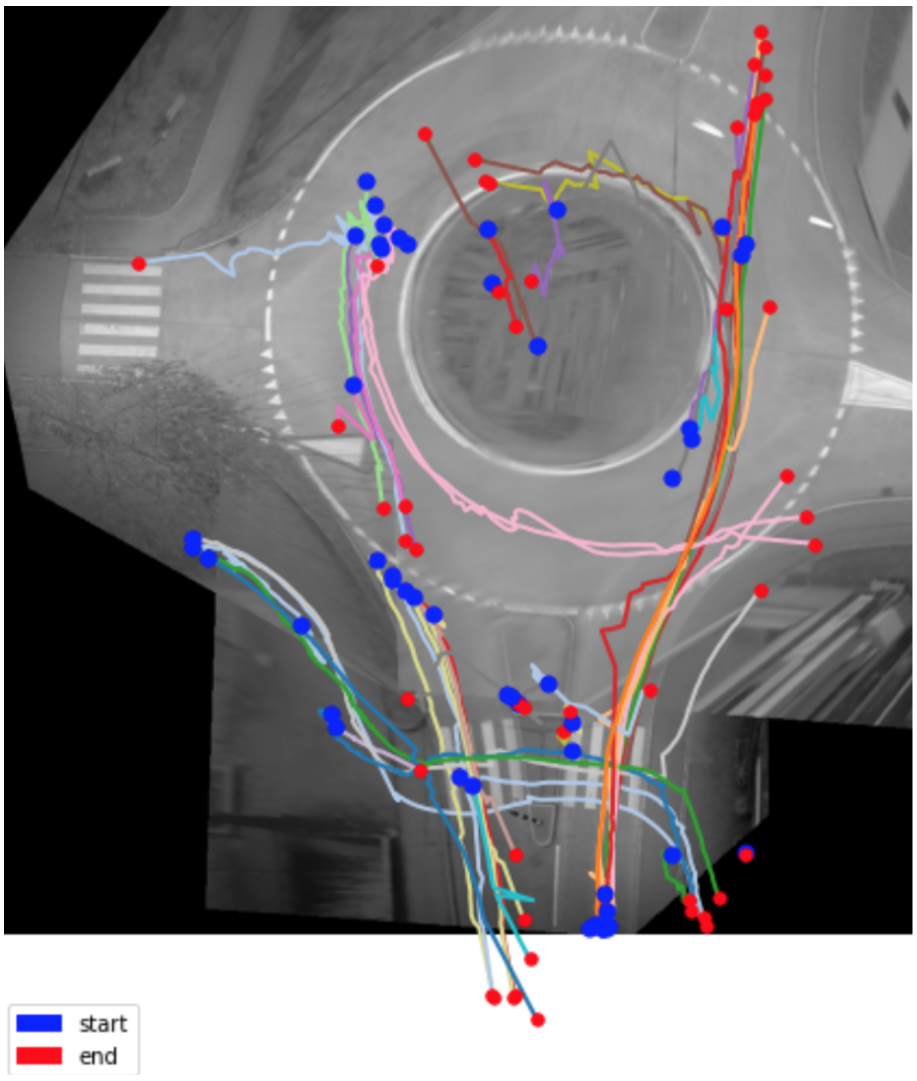


Figure 4.2: Bird's eye view of the output of the algorithm.

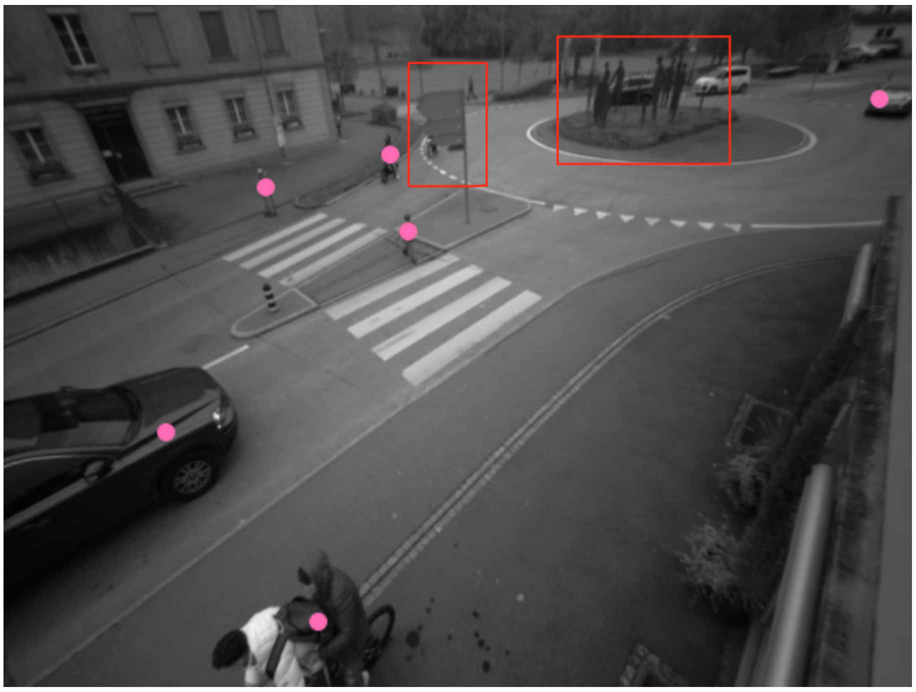


Figure 4.3: Main obstacles that characterise the urban scenario in question.

Comparing Figure 4.3 and Figure 4.2 it is possible to see some patterns in the formation of the tracks.

First of all, the presence of the traffic sign framed in Figure 4.3 creates an area where the tracks are always interrupted, as it is possible to see from Figure 4.2 at the beginning of the left lane of the bottom road. This is probably due to the fact that the detections of the vehicles disappear behind the traffic sign and the tracks are not able to follow their targets. However, by comparing subsequent frames, it is possible to notice that the occlusion lasts for very little time. Figure 4.4 shows one case of a track that is interrupted due to the fact that its target is hidden by the traffic sign.

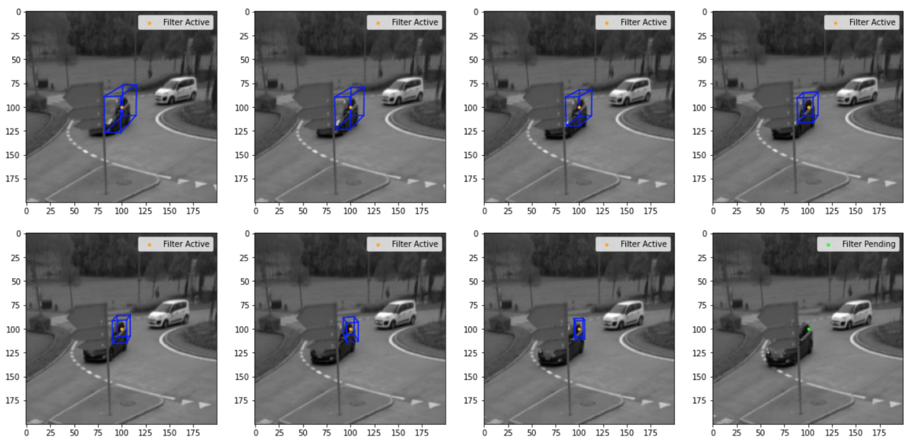


Figure 4.4: The vehicle starts to be occluded by the traffic sign. Its bounding box is squeezing.

The final subplots in Figure 4.4 show that the bounding box of the object reduces its dimensions due to the presence of the road sign that partially occludes the object, which also causes the position of the target to be shifted backward. As a consequence, for the tracking system this object is slowing down, while it is very likely that it is keeping its speed constant or that it is accelerating, given that it exited the roundabout. When the detections are no longer available, the filter predicts the next positions using a constant speed model. What has just been said can be deduced from the profile speed of the track under analysis, which is reported in the following figure:

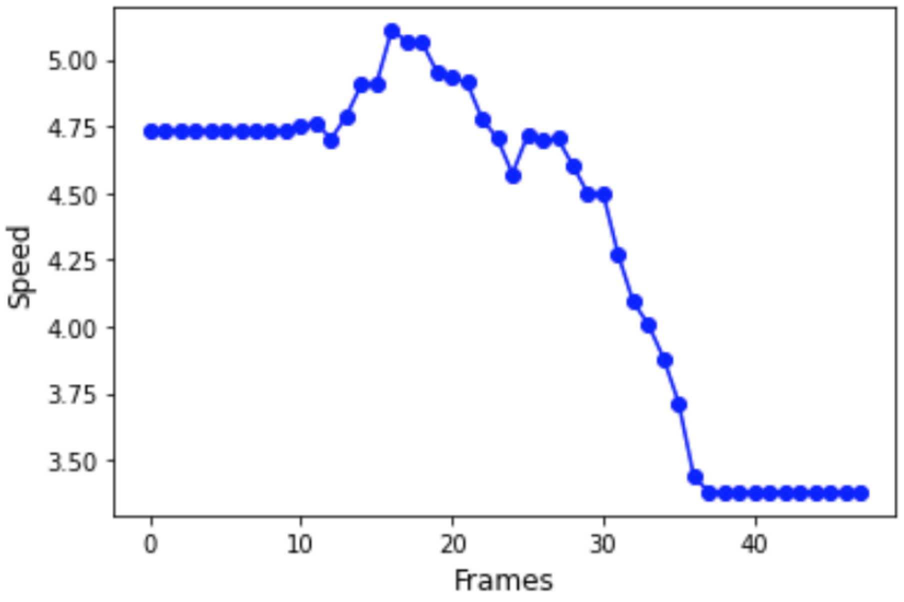


Figure 4.5: Speed (in m/s) profile of the track represented in Figure 4.4.

This plot shows the perceived deceleration phase and the constant speed phase. The constant speed model does not manage to fill the gap created by the deceleration phase, and the distance between the position predicted and the actual detection becomes large. Considering also that the Intersection over union between the track's bounding box and a generic detection must be bigger than zero to perform association, the track is not associated with the detection of the vehicle whenever it becomes available again. As a result, the track loses its target.

The other obstacle that has been previously mentioned are the statues in the middle of the roundabout, which make it almost impossible for the sensor to detect the objects behind them. As a consequence, there is almost no track in that part of the roundabout among the ones of the output.

It is thus evident that a major issue for this tracking system is dealing with occluded objects.

In addition to what has already been noted, there are other tracks that are originated by these obstacles and that can be considered "problematic". As a matter of fact, it is clear from Figure 4.2 that in the output of the algorithm there are some noisy tracks, corresponding to the ones that have both starting and ending points on the roundabout or in between the two lanes of the bottom road. Those are very likely to be *False Positive* tracks since in those spots there cannot be any road user. Later on there will be a discussion regarding this type of track.

On the other hand, the Tracker manages to smoothly track some targets: pedestrians that cross the street, vehicles that travel from the bottom road to the top road ("bottom" and "top" are referred to Figure 4.2), or vehicles that go around the roundabout and exit from the road on the right. These are some examples of trajectories that are successfully created by the tracking system.

The bird's eye view allows for a first qualitative analysis and can be used to give an overview of the results of the Tracker. Now it is better to go into details and see how the tracker behaved in problematic situations.

4.2.1. Importance of Intersection over Union Constraint

Due to the Intersection over Union constraint, a track can be associated with a detection only if the **Intersection over Union** (IoU) between the bounding box of the track and the bounding box of the detection is larger than zero. A simple reason why this constraint is necessary is that the logarithm of the intersection over union is part of the score of the track, as explained in Section 1.2.2. However, this case could be easily solved by assigning a very negative value whenever the IoU is zero. The real reason the constraint is needed is to better control the type of associations that are made. A Multiple Hypothesis Tracking system relies on the gating area to look for the next position of the track, but if the track is not associated with any detection for consecutive frames, the covariance of the filter of the track increases and so does the size of the gating area, since its dimensions are given by the level curve of the Mahalanobis distance, as shown in Section 1.2.1. The situation described is verified, for example, when an object leaves the roundabout: no detections are available anymore for the object, but the track predicts its positions for ten frames before noticing that the object left the scenario. As a consequence, the covariance of the filter explodes and the track might be associated with detections related to objects entering the roundabout. Figures 4.6 and 4.7 are examples of this type of track. The two images show the same track which associates two different targets: this happens because the original target, i.e.,the white car, is leaving the roundabout and the black car is entering almost at the same time.

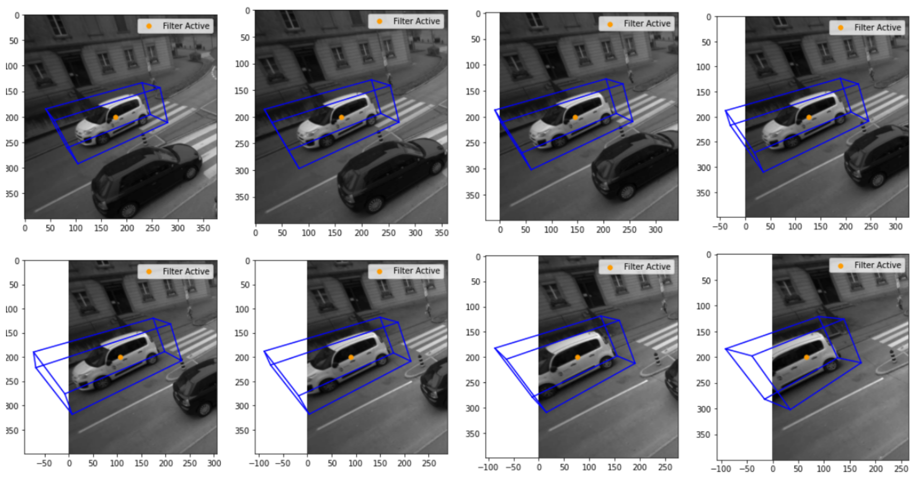


Figure 4.6: Here the track is tracking a white car leaving the roundabout.

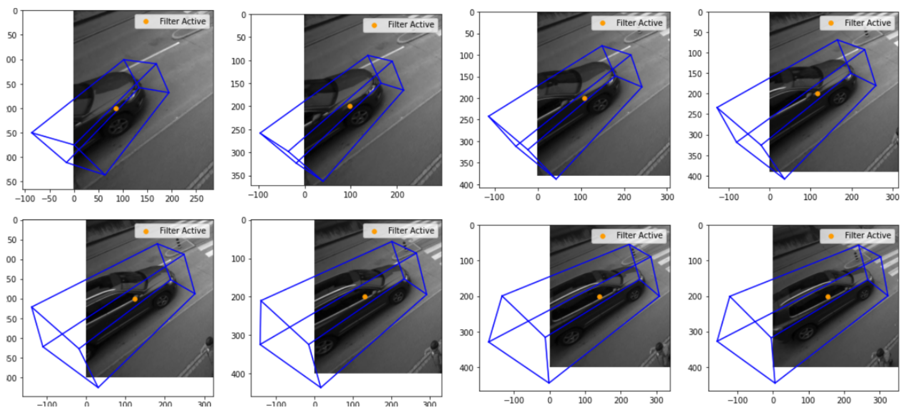


Figure 4.7: Same track as Figure 4.6, but now the target is a black car.

The gating areas are not small enough to avoid these associations, and the result is a track that switches its target (*Identity Switch*). It is clear that such a situation can be solved by deploying the IoU constraint.

The only exception is made in the case of tracks associated with fused objects. As stated in 3.2, if two tracks are associated for a long time with objects that represent two fused road users, when the detections for the targets involved in this fusion situation are again available, the IoU constraint is not applied. This is due to the fact that it is likely that the track predicts positions that are slightly distant from the real ones, and when the real ones become available again, the two bounding boxes might not intersect. Only in this specific case, having the guarantee that a detection is available again, it is possible to make an exception and avoid considering the IoU constraint.

4.2.2. Discussing Two cases of Fused Objects

It has already been stressed out that the proposed algorithm is also designed to handle situations in which two road users are detected as one. To understand if these situations are actually solved, the trajectories that the algorithm produces will be analyzed.

The first case is depicted in Figures 4.8, 4.9, 4.10 and 4.11. The figures represent a case of fused objects, as it is possible to see from some of the depicted frames. In particular, the cyclist and the pedestrian, for whom there are two separate tracks, are merged into only one detection for a certain number of frames. The algorithm manages to understand that there is a case of fused objects, as the blue dots in the corresponding frames show in Figures 4.8 and 4.10, and is also able to keep the two tracks separated after the merging is finished, as Figures 4.9 and 4.11 show.

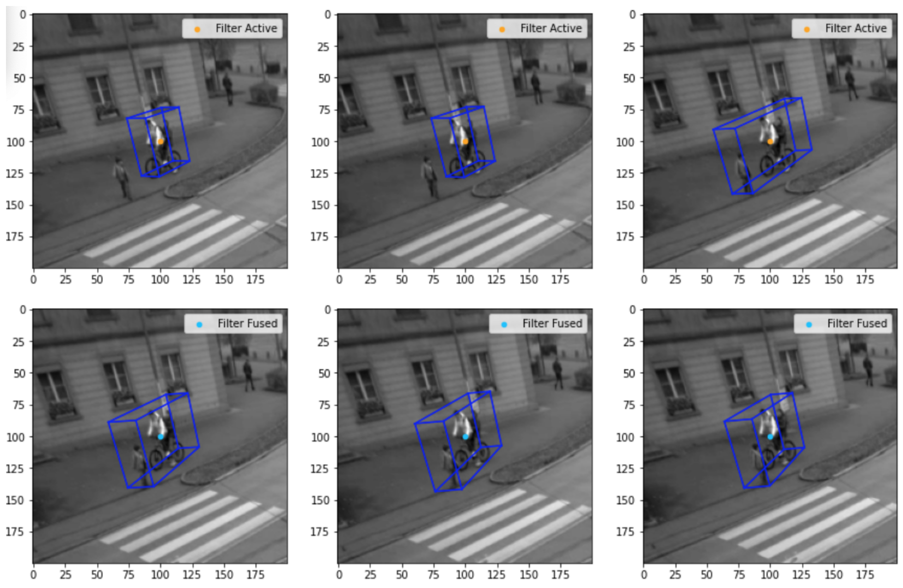


Figure 4.8: The track that is tracking the cyclist is associated with a detection that includes two road users. The detection is labeled as "Fused".

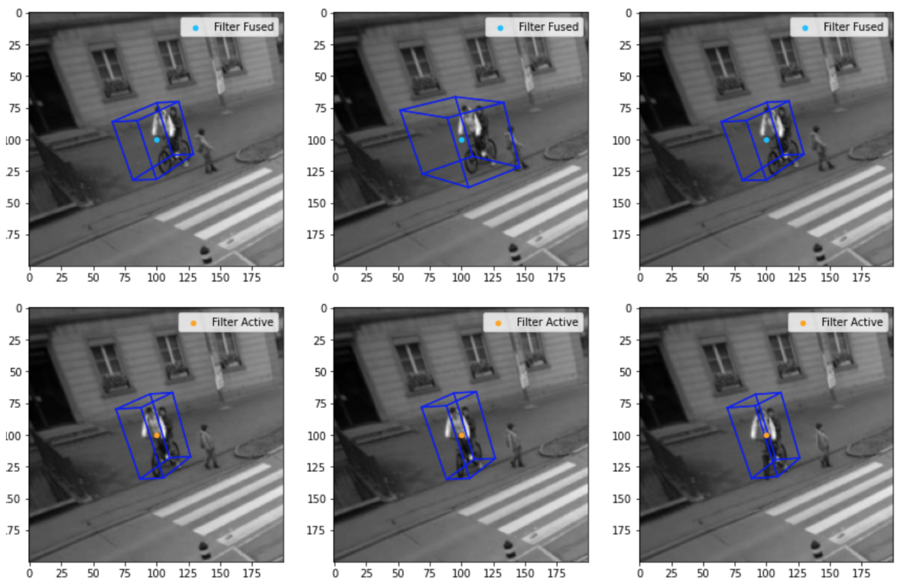


Figure 4.9: Same track as Figure 4.8. After the merging, the track goes back to its target, i.e.,the cyclist.

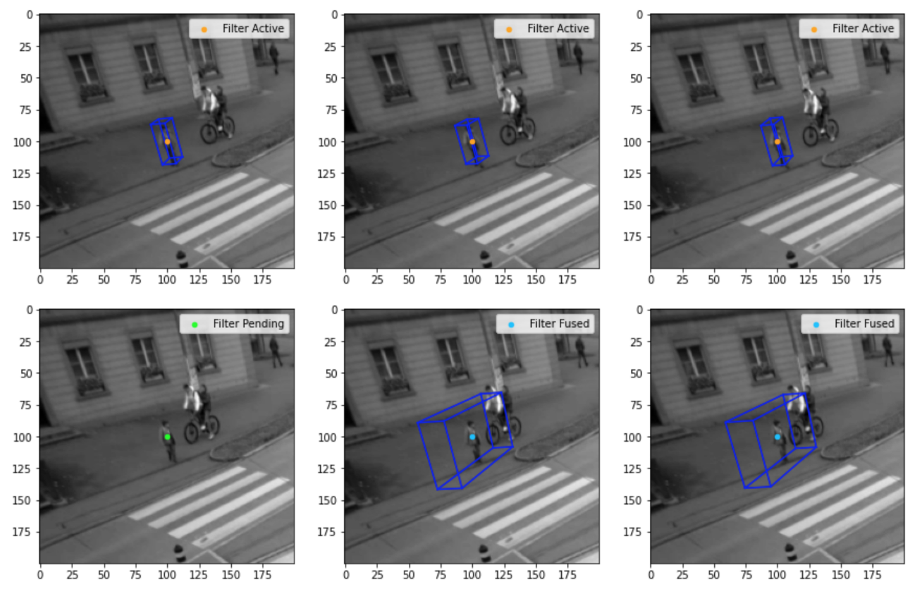


Figure 4.10: Same situation depicted in Figure 4.8, but now the focus in this picture and the next one is on the other pedestrian involved.

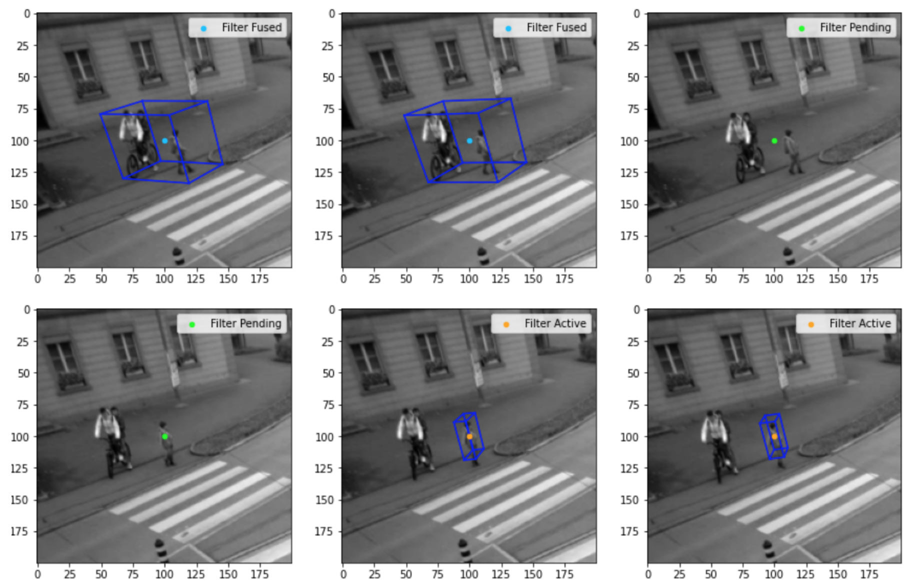


Figure 4.11: Track here is the same as 4.10. As for the other road user involved, after the merging the track goes back to its object.

The two tracks keep tracking the same object and there is no identity switch or loss of target thanks to the strategy based on Bootstrap Confidence Intervals that has been described in Section 3.2. Without this strategy, the two tracks would have been considered incompatible because they would have been associated with the same detection (i.e.,the one representing fused objects). With this strategy the algorithm recognizes the situation and after the merging it keeps the two targets separated and tracked by their respective tracks, as it is possible to see from the following plot:

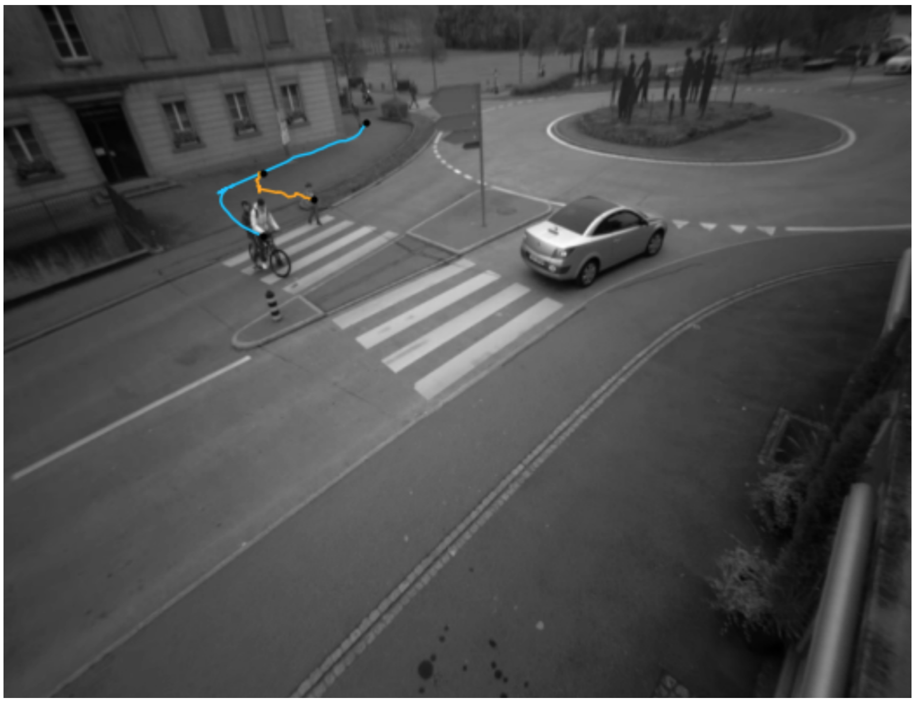


Figure 4.12: The two tracks of 4.8 and 4.10 are plotted here. The two lines represent the trajectories until the showed frame.

Despite fusion between the cyclist and pedestrian objects in several frames, the corresponding tracks are not merged, switched or interrupted. This is a significant improvement compared to a standard single-hypothesis tracker, which would have broken one of the two tracks since only one detection is available.

The case shown is not an isolated one, because the algorithm manages to solve almost all the situations in the data set of tracks associated with detections that are fused objects using this probabilistic approach.

Figures 4.13 and 4.14 show another case, which involves a scooter and two motorcycles. The proximity of the scooter and the motorcycles causes the detections for these road users to be merged. The scooter is merged with both the motorcycles subsequently, so the situation of fused detections stays for multiple frames.

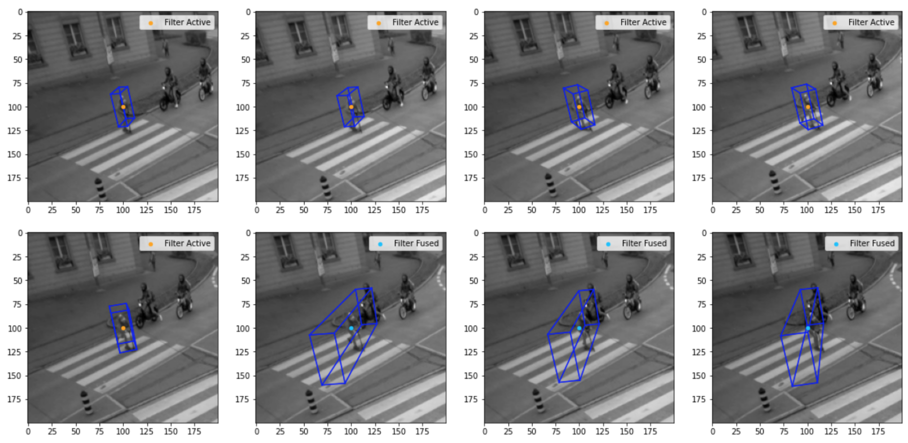


Figure 4.13: The scooter is crossing the street. The closest motorcyclist is merged with the pedestrian in a unique detection. The algorithm classifies this detection as *fused*.

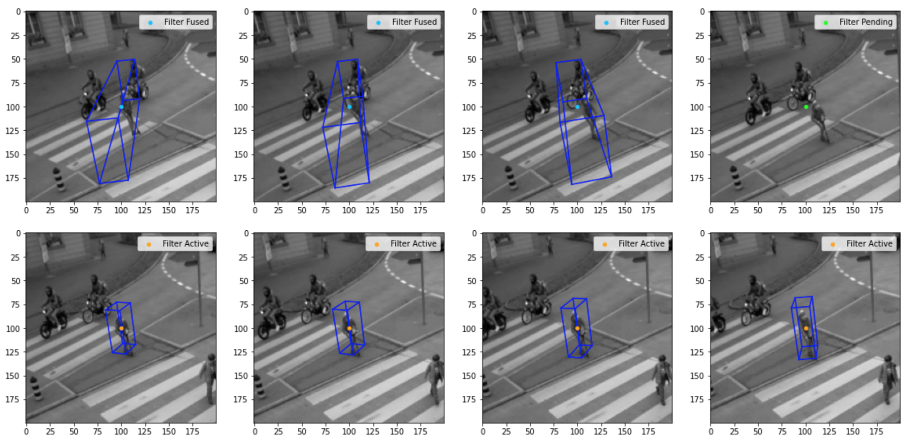


Figure 4.14: The situation of fused detections goes on also with the second motorcyclist. When a detection for the scooter is again available, the track is able to reconnnect with its target.

As it is possible to see from Figure 4.14, the track of the scooter does not lose its target after it has been merged with two motorcycles. During the frames in which the track is associated with a fused detection, the filter predicts the next position, and since this position is inside the fused bounding box, the filter does not use the information on the position coming from the bounding box, as explained in Section 3.2.

In the best global hypothesis there are three tracks for these three road users, which is a great accomplishment of the algorithm, given the fact that for multiple frames distinct detections for all three targets are unavailable. If a strategy to deal with this situation was not deployed, the results would have been worse. To prove this, the algorithm was run on the same data set without the probabilistic approach that uses bootstrap confidence intervals. The result is shown in the following figure:

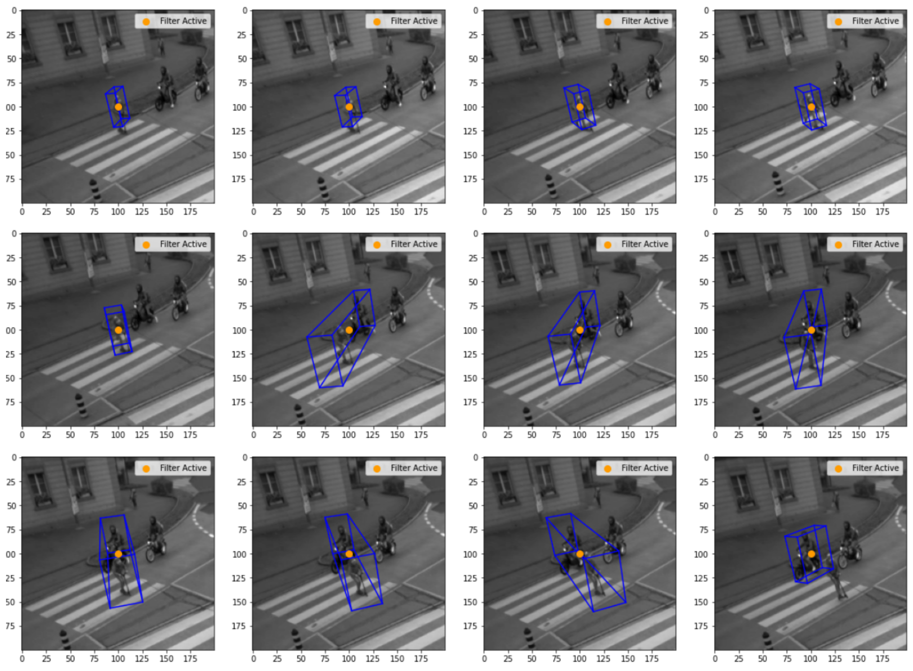


Figure 4.15: The track that refers to the pedestrian is shifted to the motorcycle after the two objects have been merged into one detection for multiple frames. This track is obtained with the baseline tracker, that is, without using the bootstrap strategy.

Initially, the track's target is the scooter, but after the merging, the target becomes the motorcyclist. As a result, a new track for the scooter will be created, and the old track for the motorcyclist will be pruned given its incompatibility with the track that is now tracking the motorcyclist. It is fair to say that this is also due to the type of score that has been designed for the tracks (Section 1.2.2), which uses only the Intersection over Union as appearance score, without considering the shape of the bounding boxes.

Regarding the proposed algorithm, Figure 4.17 shows a top view of a part of the tracks of the three road users involved in this case study, from their beginning to few frames after the fusion. From these pictures it appears that the three tracks are kept separate, but the smoothness of the tracks is affected by the absence of reliable detections, as the trajectory of the scooter (the horizontal one) shows.

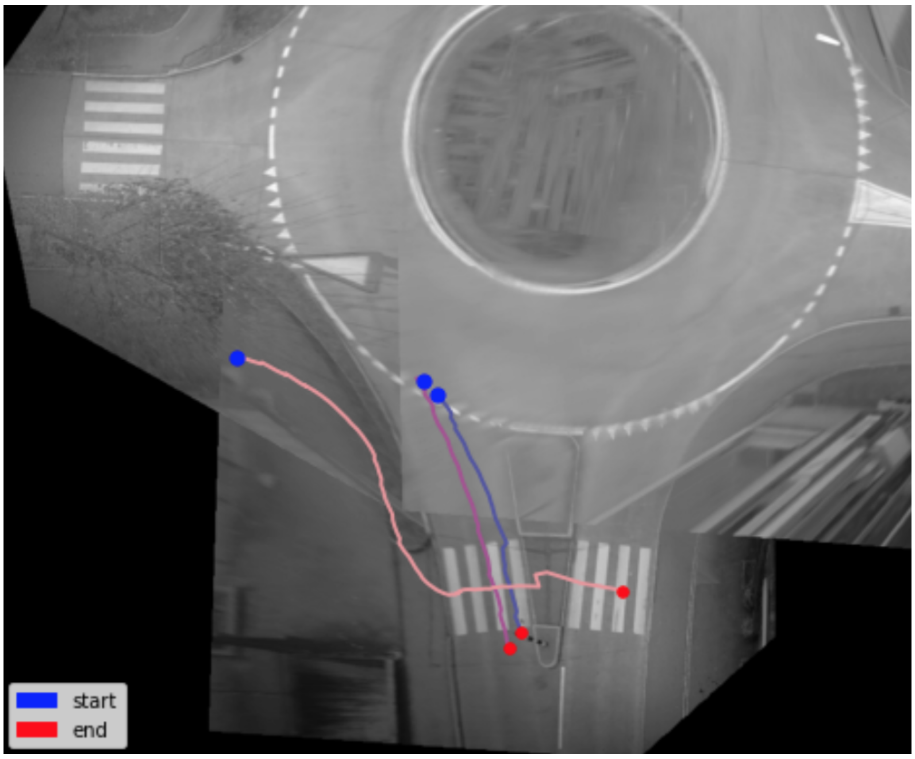


Figure 4.16: Tracks of the pedestrian and the two motocyclists. They are separated despite the presence of fused detections.

The two cases of merged objects analyzed so far are very similar and they have another characteristic in common: the detection associated with the track of the pedestrian in Figure 4.11 and of the scooter in Figure 4.14 when there are no more fused detections is the closest available in terms of euclidean distance. Thanks to the Multiple Hypothesis approach deployed in this work, the track is associated not only with the closest detection available but also with other ones, and this helps the tracker identify the best option. Looking at the tracks that have been pruned because incompatible with one of the tracks mentioned before (pedestrian and scooter), it is possible to appreciate the importance of Multiple Hypothesis Tracking in these situations: Figure 4.17 refers to the scooter, while Figure 4.18 to the pedestrian whose detection is merged with that of the cyclist.

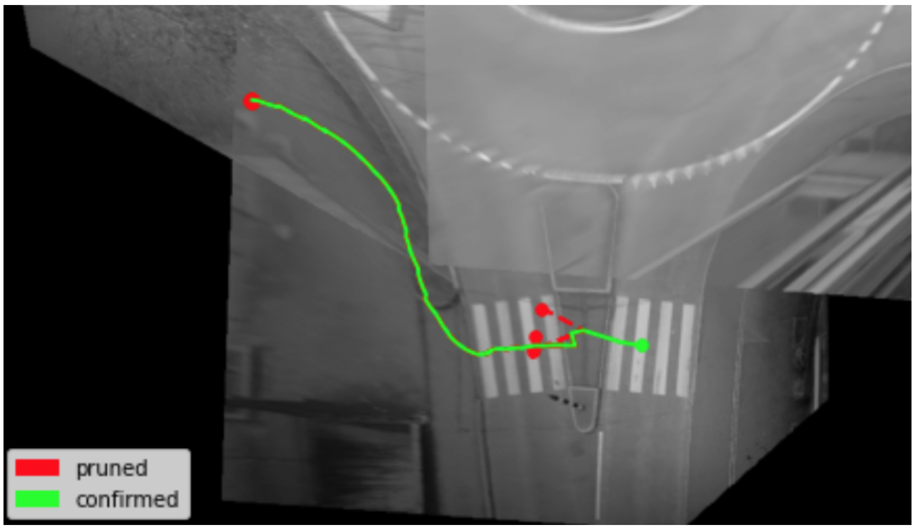


Figure 4.17: In red, tracks pruned based on incompatibility with the real trajectory of the scooter, which is plotted in green.

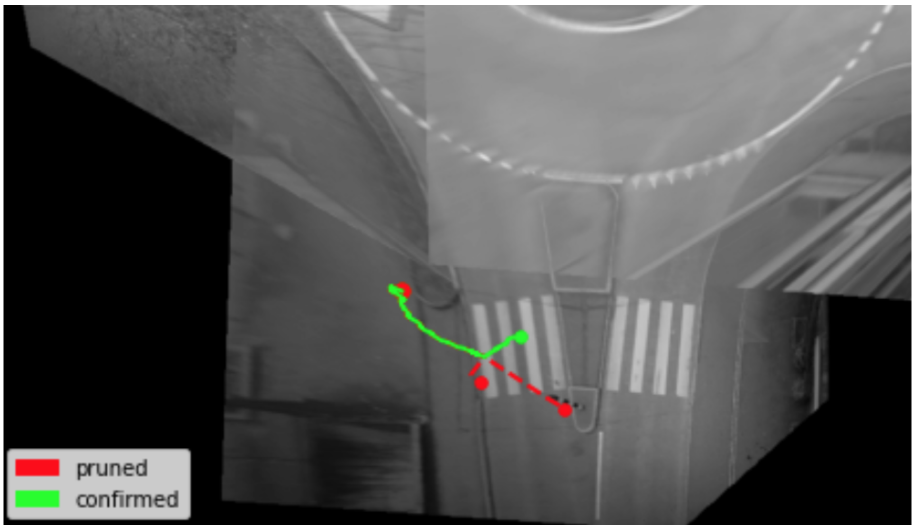


Figure 4.18: In red, tracks pruned based on incompatibility with the real trajectory of the pedestrian, which is plotted in green.

We can see that the pruned tracks depart from the track that represents the real trajectory. Each track is a hypothesis, and thanks to the procedure explained in Chapter 3, only one of these hypotheses per target is kept, based on their motion and appearance characteristics.

4.2.3. General Results of the Algorithm

In the previous section, two particular cases have been analyzed. In this section, general results will be presented and discussed, starting with a consideration regarding the distance between the associations and the tracks in comparison with a single-hypothesis tracker. The two cases presented before show how the closest detection is not always the best to be associated with the track. However, only nine tracks in the best global hypothesis are associated with a detection that is not the closest one, and overall this happens in a total of 13 frames. These digits make it clear that in most of the cases in this scenario the best detection to be associated with a track is the closest one. As a consequence, the usage of Multiple Hypothesis Tracking is not strictly necessary in most of the cases, although it is very useful in these situations of fused detections if coupled with a proper strategy, as shown in the previous cases.

As a matter of fact, the scenario in question is not a densely populated one, where multiple hypothesis tracking could be useful in more situations. However, the computational effort of the proposed algorithm is not high, since the processing of each frame requires less than a second. As a consequence, using a Multiple Hypothesis approach does not slow down the algorithm significantly enough to prefer a single hypothesis tracker, and it allows one to take into consideration more than one trajectory in order to determine the one that is most likely to represent the target.

Regarding the time complexity of the algorithm, it is possible to say that the association phase, which includes the association of detections with the tracks and the update of the score of each track, is the one that takes the most time on average: 0.91 seconds. In total, all the frames are processed in an execution time equal to 699 seconds, which corresponds to less than a second per frame on average (there are 750 frames in the data set). All these values have been computed with the **time** module in *Python*, using the function *time.perf_counter()*. The following table reports the average time deployed by the three most important phases of the algorithm:

Computational Load

Phase	Average time (s)
Association	0.91
Global Hyp.	0.02
Pruning	$2.7\text{x}10^{-3}$
Total	0.932

Table 4.2: Summary of computational load of the proposed algorithm. "Global Hyp." stands for computation of the best global hypothesis (Chapter 3).

Now it is time to analyze more broadly the impact of the usage of bootstrap confidence intervals, and the impact of the strategy to detect fused objects. In particular, it is possible to produce a *confusion matrix* to quantify how many cases of merged objects are correctly identified by the algorithm. To produce this table, all the cases in which bootstrap confidence intervals have been computed have been analyzed and manually labeled as cases of fused or not fused detections.

Confusion Matrix

	True Labels	
Predicted	**Not Fused**	**Fused**
Not Fused	64	8
Fused	122	76

Table 4.3: Confusion Matrix for the detections related to merged objects.

From the values reported in the table it is possible to compute the **Accuracy** and **Recall** of this algorithm in detecting merged objects:

$$\text{Accuracy} = \frac{tp + tn}{N} = 0.5185 \qquad \text{Recall} = \frac{tp}{tp + fn} = 0.904 \tag{4.1}$$

Where tp stands for *true positive*, tn stands for *true negative* and fn stands for *false negative*. N is the total number of cases.

The recall is very high, meaning that the algorithm detects all the problematic cases. However, the Accuracy of the algorithm in detecting fused objects is very low. Looking in a better way at the cases in which these bootstrap confidence intervals are used to assess if two objects are fused or not, it is possible to find two main reasons that cause the accuracy to be this low:

1. There are cases of multiple detections for the same object. These multiple detections create issues because they originate two different tracks for the same target, with bounding boxes that are smaller than the dimensions of the object. These tracks will eventually be associated with the same detection, since usually the situation of multiple associations lasts for a few frames. As a consequence, bootstrap confidence intervals will eventually be used to understand if the two tracks represent two objects that are merged, and it is very likely that the answer based on those intervals would be positive given that the size of the bounding boxes is smaller than the size of the

object.

2. It happens that some noisy detections belonging to stationary objects become tracks. These tracks do not refer to any real target (*False Positive* tracks) but they can be associated with detections that actually belong to other tracks. In this case, the bootstrap confidence interval may label this situation as a fused one, allowing the *False Positive* tracks to remain in the scenario for more frames without being pruned.

These two situations are common in the data set and can be problematic. In general, this means that the real problem to face is that these bootstrap confidence intervals are used even in situations that do not require them to be computed. The bootstrap confidence intervals only rely on one piece of information: the volume of the bounding boxes. It is therefore clear that more than the strategy itself, what must change is the number of times and the type of cases in which it has been used. Up until now, the only condition to be satisfied in order to use these intervals is that two tracks have to be associated with the same detection. This condition is not strict enough, as the following examples will show. Figure 4.19 shows a track that is formed by noisy detections related to a traffic sign:

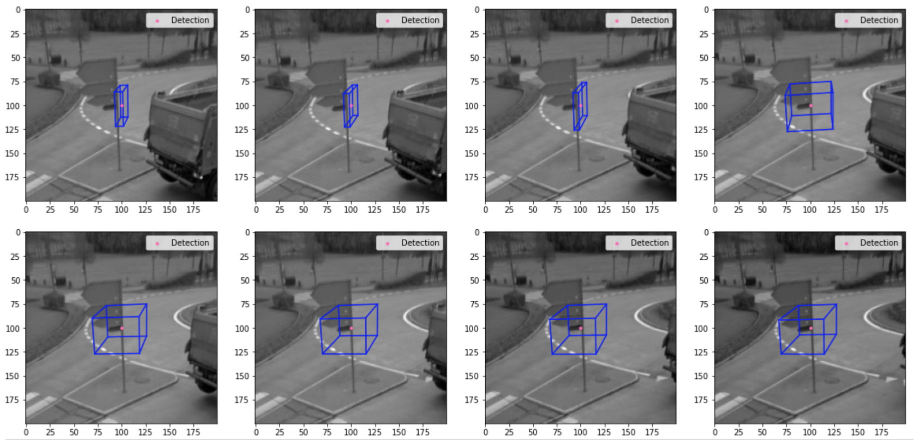


Figure 4.19: Track originated by fallacious detections that are present for multiple frames.

This track is later associated with a detection coming from a truck, and given the dimensions of the bounding box this is considered as two fused objects, as the following figure shows:

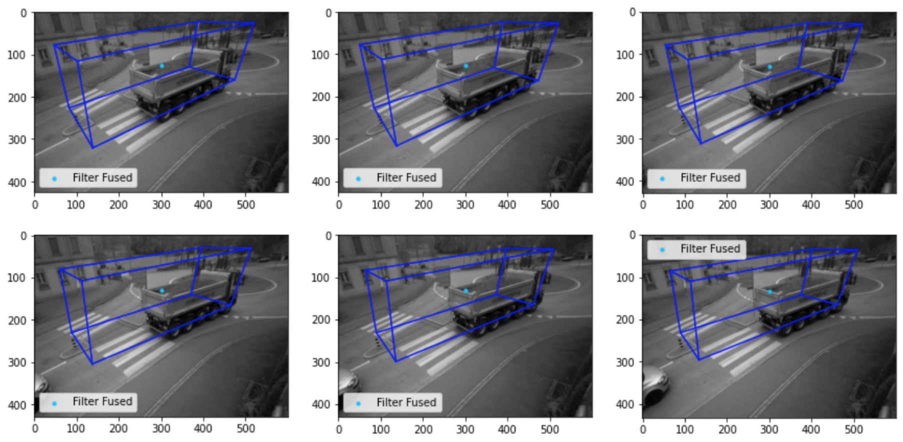


Figure 4.20: The false positive track of figure 4.19 is associated with the detection representing the truck, which is labeled as fused.

This track belongs to the output of the tracking system even though it does not represent a road user. The association with the detection of the truck could have caused a switch from the traffic sign to the truck, but this luckily did not happen in this context.

To conclude, the strategy with the bootstrap confidence intervals helps to solve all the cases of fused detections but it suffers from producing too many false positives. This is due to the fact that the comparison of the volume with the confidence interval of the track is used too much. An improvement that can be made to this work is to consider the introduction of constraints to understand when it is safe to use bootstrap confidence intervals and when they should be avoided since the two tracks involved do not refer to two distinct road users but rather to the same object or to a stationary object (not a road user).

On this data set the advantages brought by this strategy are significant. However, the approach can be improved to make the algorithm even more efficient in discriminating the various cases of fused detections from cases of erroneous association.

To conclude the analysis of the results of the algorithm, it is important to mention that for 30 targets, the number of tracks that the algorithm produced is 50. Among these tracks, there are four cases of *Identity Switch*, that is, tracks that change their targets, and eleven cases of *False Positive* tracks, that is, tracks that do not have a road user as a target. Figure 4.21 shows an example of *False Positive* track, which is originated by consecutive detections of a stationary object that does not represent any road user.

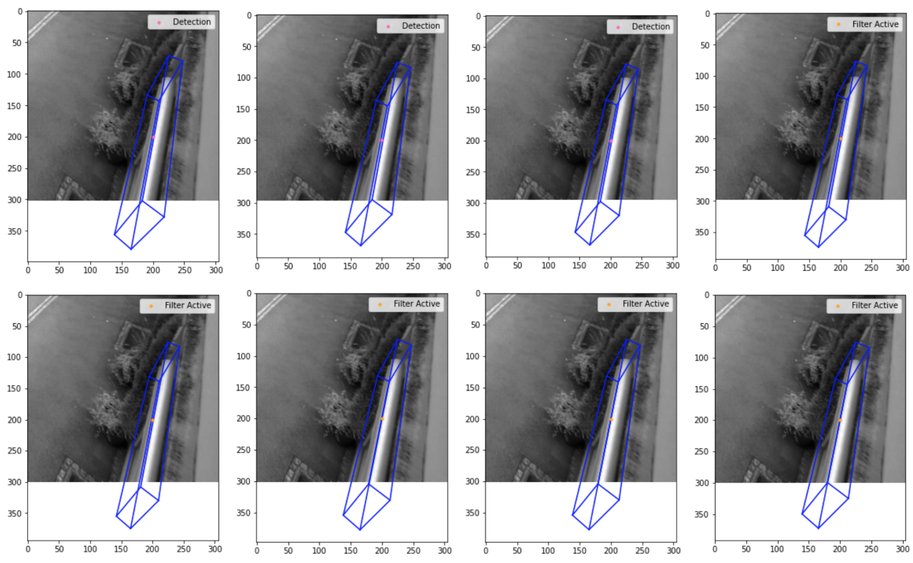


Figure 4.21: Example of *False Positive* track

Moreover, from Figure 4.22 it is possible to see all the *False Positive* tracks from above. It is clear from this picture that these tracks are the result of the presence of obstacles, like the statues on the roundabout and the traffic sign.

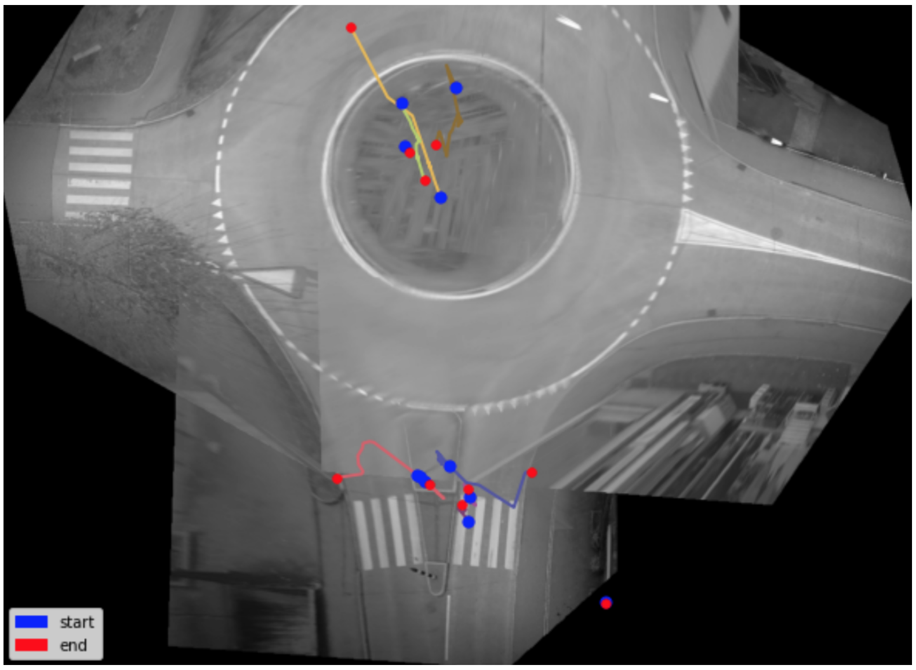


Figure 4.22: Bird's eye view of all *False Positive* tracks

As it is possible to see in Figure 4.22, the false positives mostly stay in place. The position variance is small compared to other tracks that cross more or less the entire field of view.

This ideally makes it possible to filter out these tracks in the after-filtering stage.

In addition to this, there are only two targets that do not have a corresponding track: both of these targets are pedestrians that are joined in one detection with another pedestrian for most of their lifetime, except for two or three frames. The main reason for the absence of a track for these two pedestrians relies on the detections of those targets, which is not available for most of their presence in the scenario.

To conclude, it is possible to say that the algorithm performs well on the given data set and is able to solve cases of fused detections while keeping the number of identity switches very low. The main weak spot is the extreme usage of the strategy that uses bootstrap confidence intervals to solve cases of fused road users, which leads to some noisy tracks interfering with tracks related to real targets. In general, the algorithm has promising results, and it can be tested on other urban scenarios.

5 | Additional Experiments

In this chapter there will be a discussion about other strategies that have been initially implemented to solve the tracking problems discussed in Chapter 2, before the introduction of the strategy based on bootstrap confidence intervals, which is part of the proposed algorithm. As such, results related to the algorithm deploying these alternative strategies will be shown. In every section it will be clearly stated which version of the algorithm to refer to. Some versions of the algorithm will be used to analyze a certain strategy, and others will be used to make comparisons with the proposed algorithm.

Last but not least, there will also be a comparison between two different motion models that have been tested: **Costant speed model** and **Costant Acceleration model**.

5.1. Speed and Acceleration Limits

One approach that has been considered to address problems caused by false or fused detections is the introduction of constraints in the association phase. These constraints were thought to avoid any type of unfeasible association, and were mainly focused on keeping under control the speed and acceleration of each track. As a consequence, the tracking system used in this section does not deploy any strategy to detect fused objects. This is mainly due to the fact that these constraints are thought to prevent the tracks from associating with numerous detections, and thus the assumption is that the detections representing fused objects will not be associated with the tracks, given the fact that they are often misleading both for the size of the bounding box and for the center of the bounding box, that is, the possible next position of the track.

Last but not least, the Kalman Filter used by the tracking system under analysis in this section deploys a constant speed motion model to create the trajectories.

5.1.1. Reasons behind Introduction of Kinematic Constraints

In Chapter 4, it has been mentioned that without the implementation of a strategy for detecting fused objects, the tracker would very likely create tracks that lose or switch their targets in those situations. In particular, Figure 4.15 shows one of these cases. It is possible to analyze the acceleration profile of such a track, by plotting it as it evolves across frames:

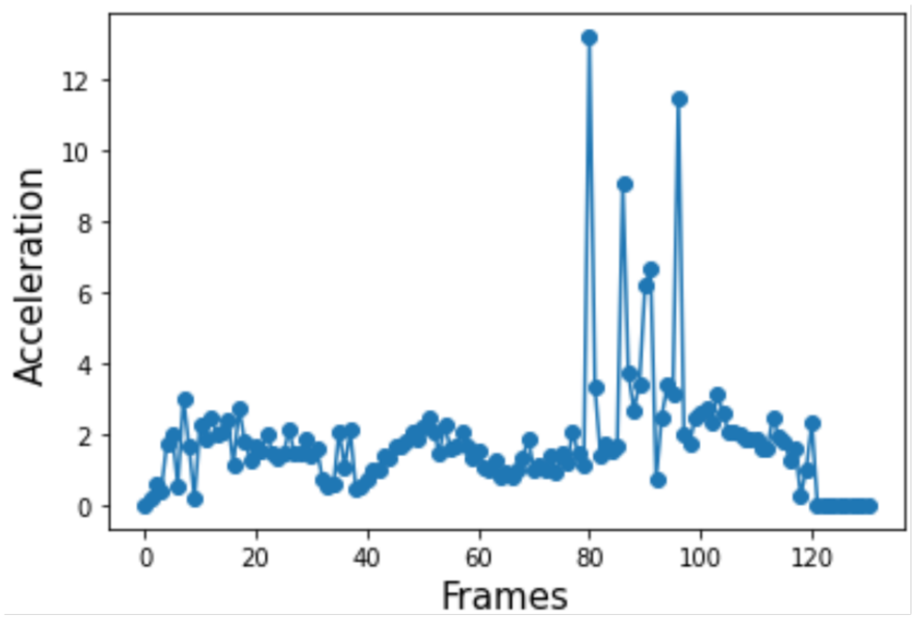


Figure 5.1: Acceleration (in m/s^2) of track in Figure 4.15.

The figure shows that the track has an unfeasible acceleration of more than 12 m/s^2 in just one frame.

This is not the only case that led me to think that kinematic constraints could be useful. Another one is represented by a track that shifts its target from a traffic sign to a vehicle. The track is shown in Figure 5.2 and it represents a problem that is not completely solved in the proposed algorithm as well, as discussed at the end of Section 4.2.3.

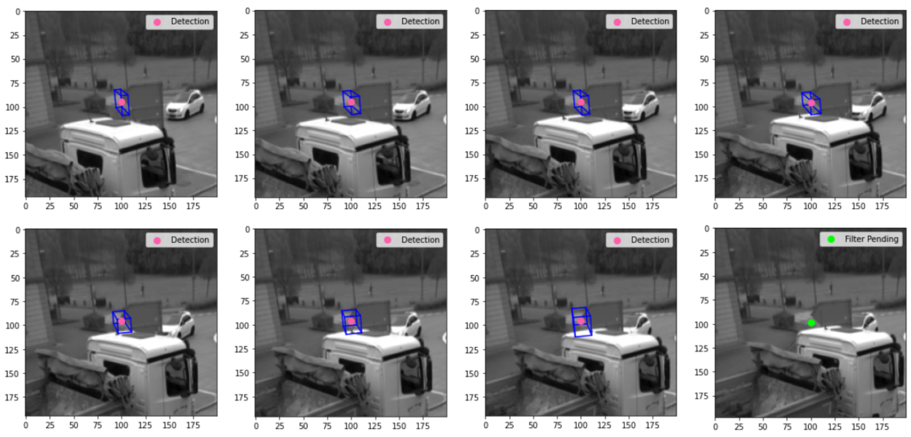


Figure 5.2: Beginning of the track represented by some noisy detections in correspondence of a traffic sign.

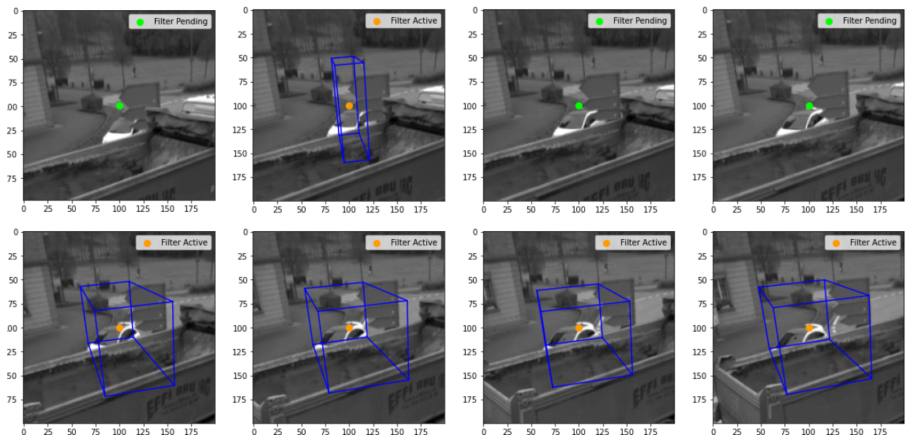


Figure 5.3: The same track of Figure 5.2 now shifts to a white car that was occluded before.

Image 5.2 shows that initially the track is a collection of erroneous detections that happened to be in consecutive frames and thus to be considered like a real target. In particular, the presence of a big vehicle like a truck and a stable obstacle like a traffic sign gave origin to these false detections. The main problem is that, as emerges from Figure 5.3, the track is then joined with a white car. This is a result of the fact that the detections from the sign move downward in the image, originating the subsequent problem of the union of a real target (the white car) with a false alarm. Nevertheless, it is clear that in only 0.08 s, which is the time difference between two consecutive frames, no object can move from the traffic sign to the car. Looking at the plot of speed of this track (Figure 5.4), it is possible to notice that between frames 20 and 30 the value of speed increases dramatically: from approximately 0 m/s to more than 3.5 m/s in just 0.08 s, giving an acceleration of around $4g$, not feasible in a traffic context.

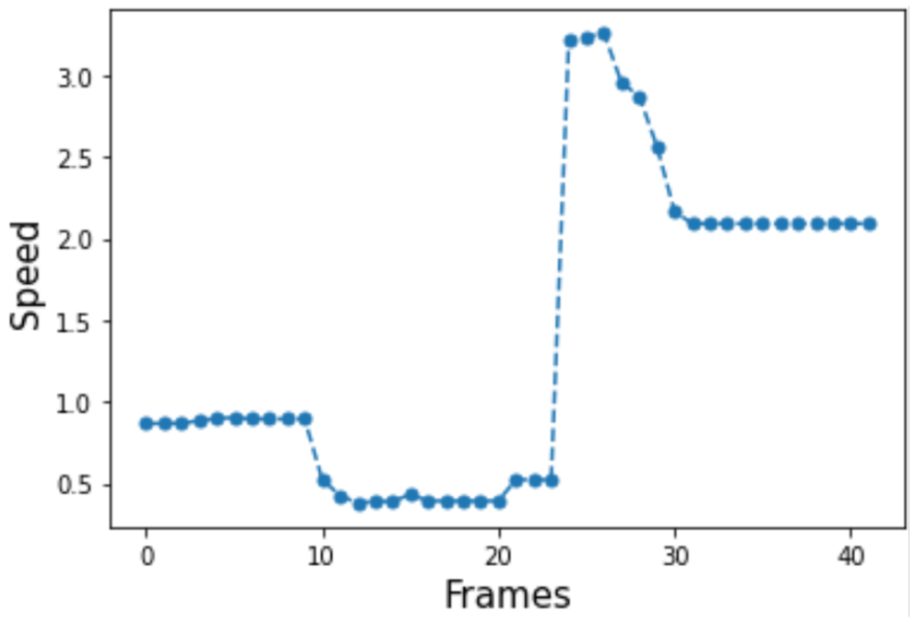


Figure 5.4: Profile of the speed (in m/s) of the track reported in figure 5.2 and 5.3.

A condition on the speed of the track may help to avoid this unfeasible behavior.

5.1.2. Results

To solve the problems discussed above, speed and acceleration limits have been introduced. In particular, the following approach was proposed: whenever a detection is in the gating area of a track, the acceleration and speed the track should have to reach the new possible position are estimated. If the speed is above 15 m/s or the acceleration is above 6 m/s^2 then the association must be discarded.

With these constraints, the number of times a track has been associated with two or more hypotheses is only 4. In total, the speed constraint was not satisfied 620 times, while the acceleration constraint was not respected 541 times.

The constraints are effective in contrasting the formation of problematic tracks like the one shown before (Figure 4.15). On the other hand, a huge drawback of this approach is that some tracks may be interrupted. This can happen with large vehicles because the detections of these targets can shift from the back to the front of the vehicle due to their sizes. As a result, the tracks of big vehicles like trucks or vans might be interrupted. Figure 5.5 is an example of a large vehicle having detections from two consecutive frames that are too distant, causing the correspondent track to be interrupted.

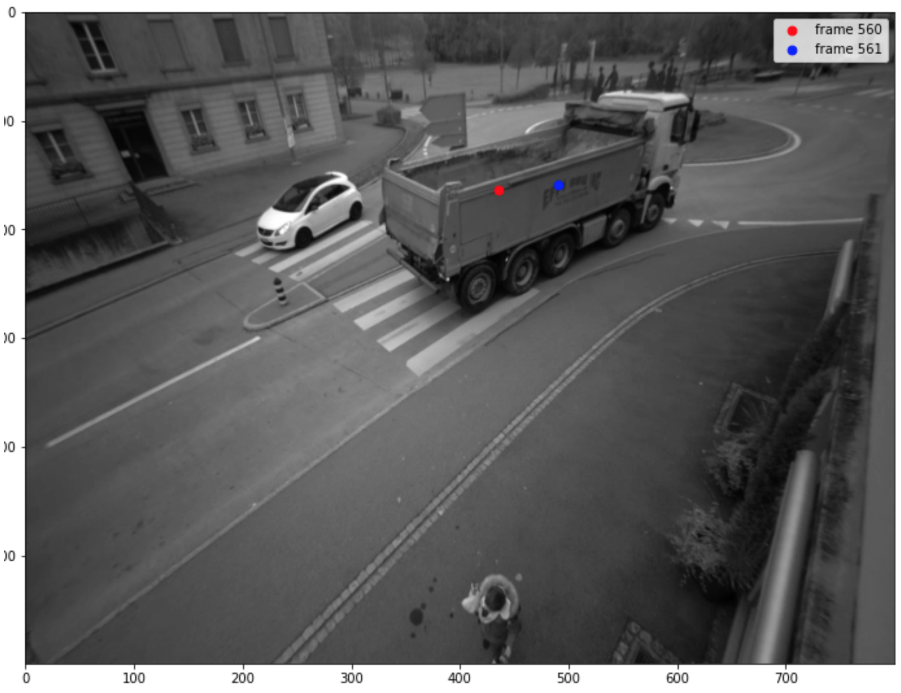


Figure 5.5: Two consecutive detections for the same vehicle.

This situation is not so common but it occurs at least once for almost all big vehicles in the roundabout. Figures 5.6 and 5.7 show another case of a track being interrupted because of the shift in detection related to the target (even though in the case depicted in the two figures, the large distance between the camera and the target might have played a role).

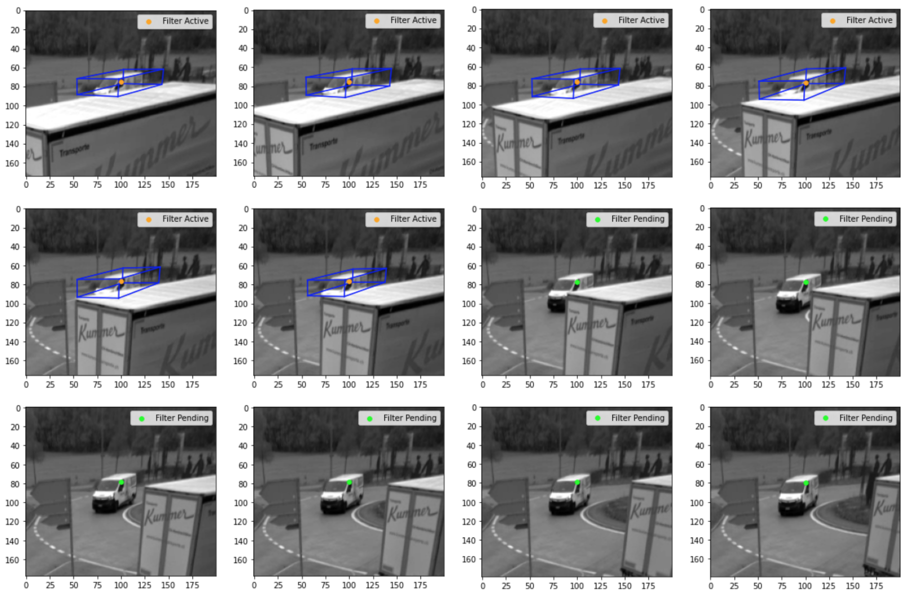


Figure 5.6: Track of the white van finishes abruptly.

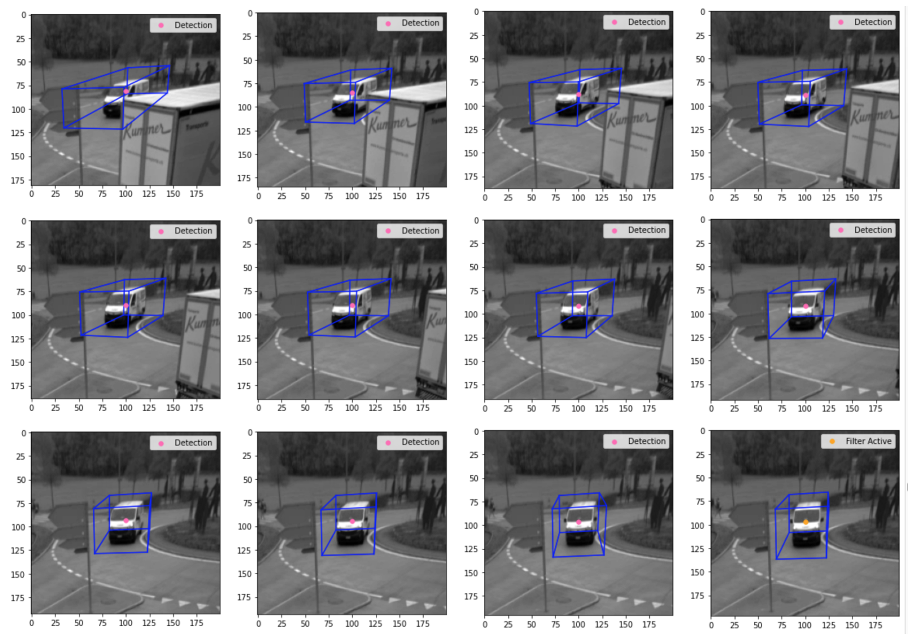


Figure 5.7: Another track for the white van is started and will continue until the van will leave the roundabout.

To summarize, the introduction of speed and acceleration limits helps to avoid unfeasible associations that could lead to unfeasible values for speed and acceleration of the tracks, but makes the algorithm way less flexible. The number of tracks created is now 81, with a lot of fragmented tracks, whereas it was 50 without constraints. This is mainly due to the fact that the detections are noisy. As a consequence, by not allowing the Kalman Filter to receive this noisy data, the algorithm is not able to filter out the noise and the estimation of speed and acceleration based on these noisy data becomes fallacious. It is possible to say that the tracker with constraints is a model that is less able to capture the variability in the data, and thus it cannot perform in a proper way its task of tracking the objects in the given scenario.

Moreover, the introduction of these constraints was motivated by looking at cases coming from a specific data set. As a consequence, even if some tracks might be improved thanks to these new constraints, the approach is not guaranteed to be valid with another data set. In other words, in a new data set, the problems related to unfeasible values for speed and acceleration might not be present at all or might be very rare. This means that these constraints cannot be considered helpful in a generic tracking system since they have been introduced as a strategy to solve some problems detected when testing the MHT system on the given data set. However, it was useful to test this approach in order to understand its advantages and disadvantages.

To conclude, it is better to avoid the introduction of this type of constraints in a tracking system given the fact that they are very likely to make the model less flexible and less able to adapt to the data.

5.2. Comparison of Motion Models

In this section the differences between the two motion models described in Section 1.3 will be explored, with a focus on the tracks that the two motion models produce. In particular, there will be a distinction between the case in which the speed and acceleration limits are deployed and the case in which they are not. The assumption is that in the case where the kinematic constraints are not used, the two motion models should behave very similarly, while in the case with speed and acceleration constraints the outcome is more unclear.

5.2.1. Case with kinematic constraints

The first case that will be analyzed is the one with kinematic constraints. This means that the two tracking systems that will be compared in this subsection use a constant speed motion model and a constant acceleration motion model respectively, but both deploy the kinematic constraints discussed in Section 5.1 and thus neither of them uses the strategy to detect fused objects based on bootstrap confidence intervals.

One major difference that can be detected when comparing the tracks produced by the two different models is that with the constant acceleration model there are more cases of fragmented tracks, probably due to the presence of the speed and acceleration limits that, by preventing some associations, do not allow the Kalman Filter to filter out the noise in some data.

As an example, two tracks produced by the tracker that uses a constant acceleration model will be analyzed. These two tracks are abruptly interrupted even though some detections for the underlined targets are still available. In particular, the constant speed model did not experience fragmentation in correspondence of the two tracks that are reported below.

1. The first case is that of a white van in the middle of the roundabout. Figures 5.8 and 5.9 show that the tracker does not continue tracking the target but stops, probably because either the speed or the acceleration limit are not satisfied. As a result, the Kalman Filter predicts the next positions only by using the constant acceleration model, which gives the vehicle a larger speed than in reality. The plot of the speed of the van in Figure 5.10 shows that the speed follows a trend that is typical of a constant acceleration model, even though before starting this trend it is clear that

the van was not accelerating.

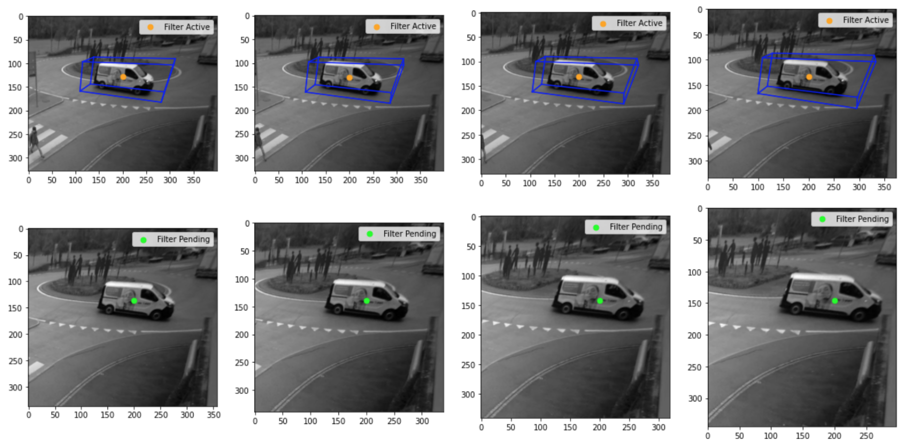


Figure 5.8: The track is interrupted even if the vehicle did not leave the roundabout.

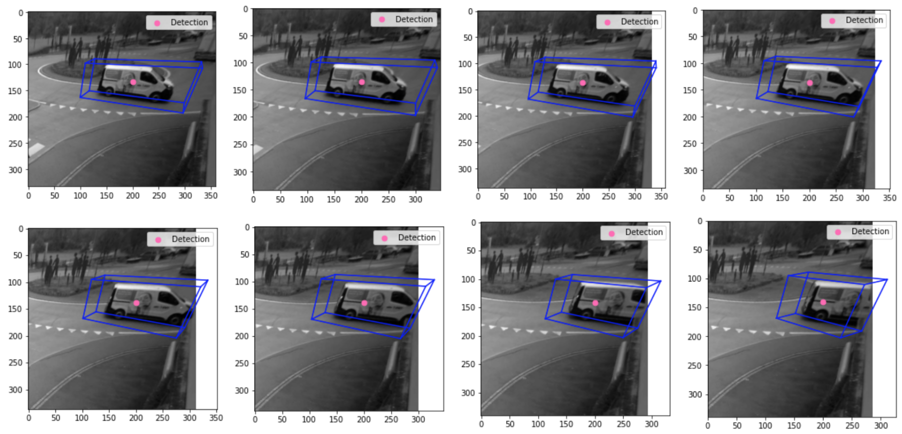


Figure 5.9: A new track is started for the same target of 5.8.

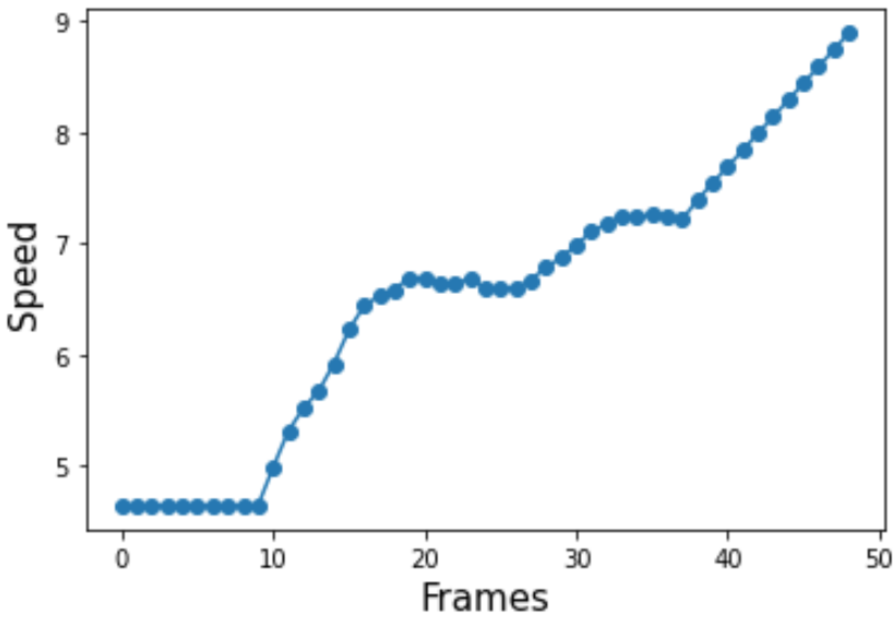


Figure 5.10: Speed (in m/s) of the track in Figure 5.8.

2. Same scenario as previous point, also in this case one track is interrupted (Figure 5.11 and 5.12) and a new one is started (Figure 5.13)

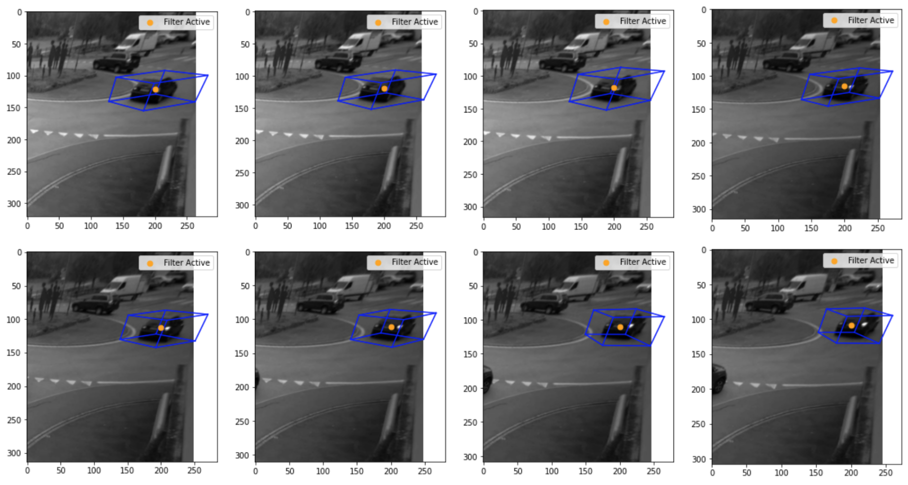


Figure 5.11: Track of a black car

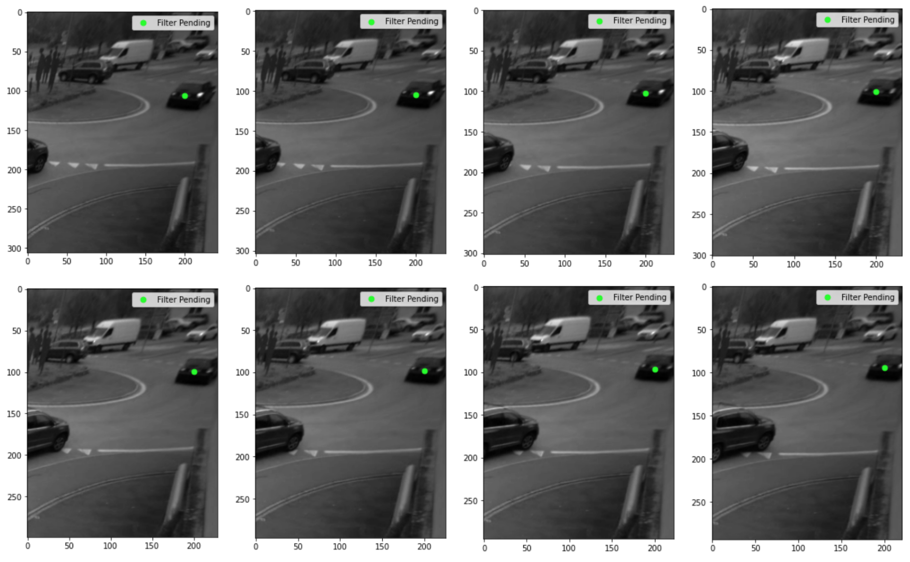


Figure 5.12: End part of track in Figure 5.11. The track is interrupted, but some detections are still available, as the next figure shows.

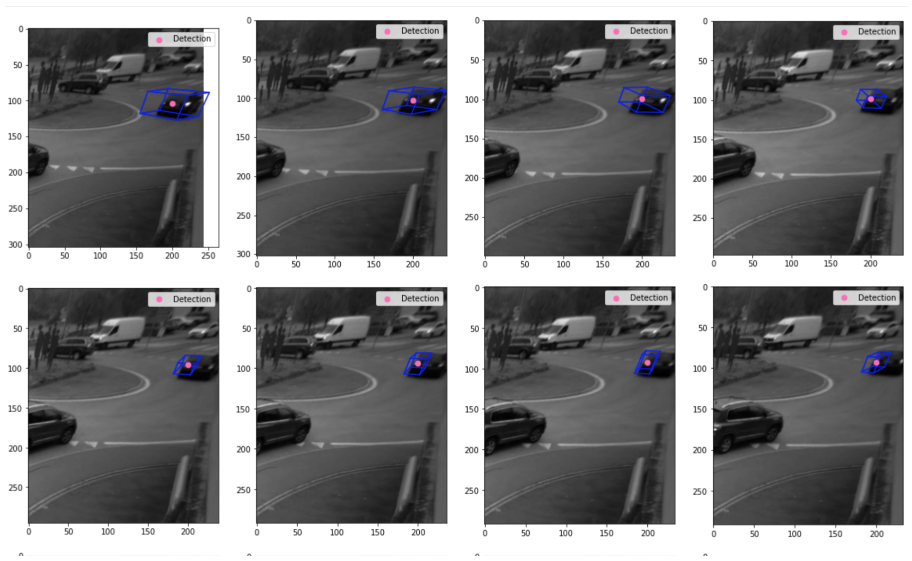


Figure 5.13: New track but same target of the previous two pictures.

This frequent fragmentation in the tracks is usually not observed in the tracker that uses a constant speed model. On the other hand, if the detections are not available because the object is occluded or two objects have been merged, the same situation may also happen with the constant speed model. This is the case of the scooter that has been discussed previously. The situation is depicted in Figures 5.14 and 5.15. From the plot of the velocity of the track (Figure 5.16), it is clear that the acceleration phase is interrupted in favor of a constant speed trend, which causes the track to lose the target.

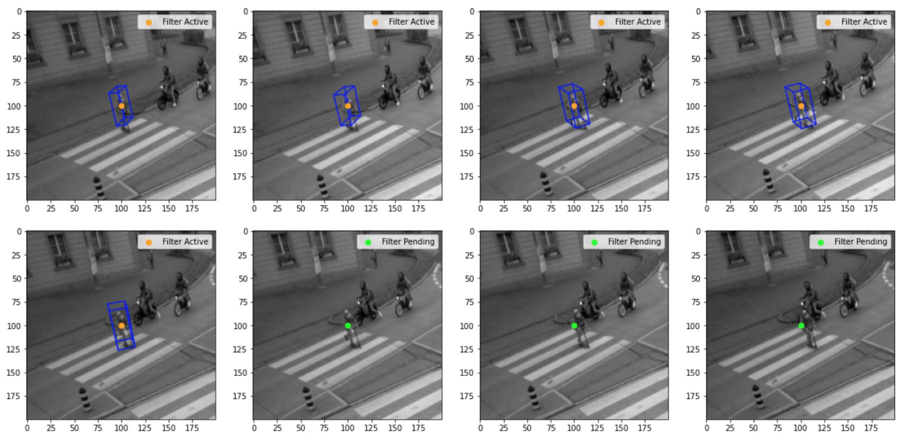


Figure 5.14: The track becomes pending because the detection merges two objects and thus does not respect speed and acceleration limits.

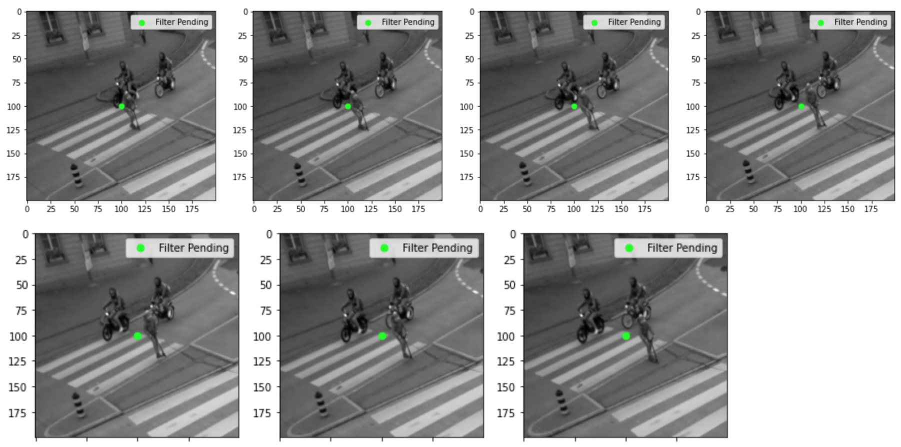


Figure 5.15: The constant speed model tries to catch up the target but it underestimates the speed of the object.

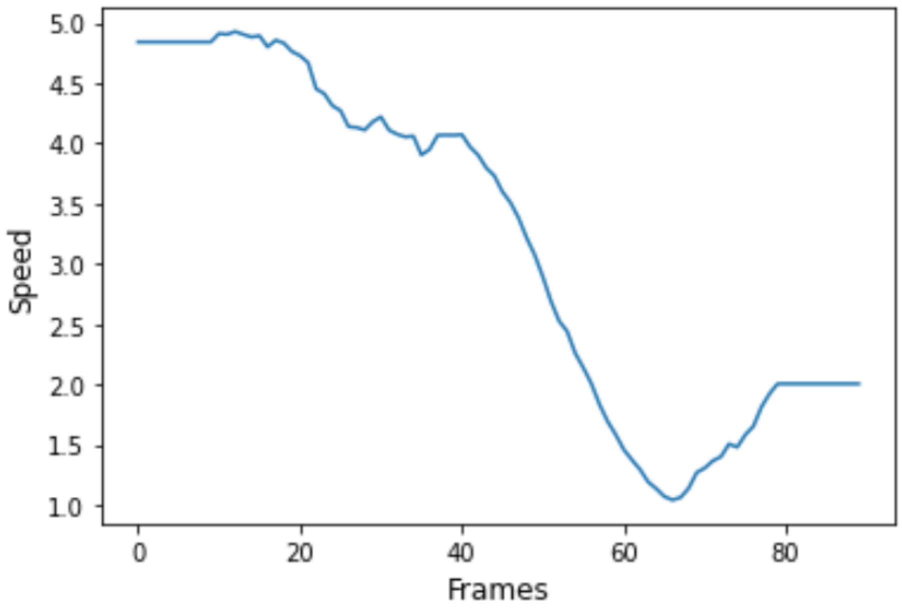
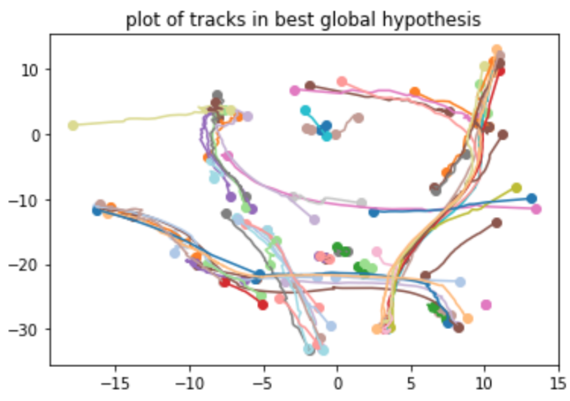


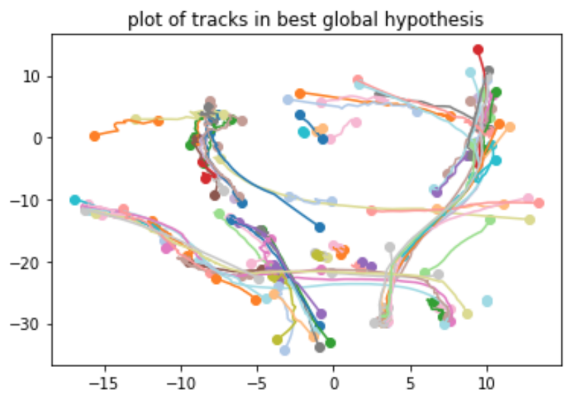
Figure 5.16: Plot of the speed (in m/s) of track in Figure 5.14. The acceleration phase at the end is interrupted, the speed becomes constant.

Both the models cannot handle a significant number of frames without individual detections associated with the track because the two motion models deployed are very basic, particularly when it comes to describing the motion of vehicles in a roundabout, where the speed and acceleration of the vehicles can change a large number of times. Between the two trackers, the one with a constant speed model seems to perform better than the one with constant acceleration when kinematic constraints are deployed. In the former, the tracks are interrupted either due to the presence of obstacles or to the fusion of two or more targets. In the latter, the fragmentation is slightly more frequent, as can be seen by comparing the number of tracks created by the tracker using the two motion models: 81 for the constant speed model and 90 for constant acceleration.

This difference emerges also when comparing the plots of the tracks in the best global hypothesis with a top-view plot: Figure 5.17a for the constant speed model, Figure 5.17b for the constant acceleration model.



(a) constant speed model



(b) constant acceleration model

Figure 5.17: Comparing tracks of two motion models from above

5.2.2. Case without kinematic constraints

Now the two tracking systems that will be compared are the ones that use the strategy based on bootstrap confidence intervals to detect fused objects and that do not deploy any kinematic constraints. The motion models of the two tracking systems are, as explained before, the constant speed model and the constant acceleration model.

The algorithm proposed in this work will thus be compared with an almost identical version, with the only difference being the type of motion model implemented for the Kalman Filter. The main reason why this different motion model was tested, despite the fact that the proposed algorithm manages to solve most of the problematic cases that characterize the given data set, was to see if a different motion model could perform better in cases of occluded objects.

As discussed at the beginning of Section 4.2, the traffic sign and the statues in the middle of the roundabout (Figure 4.3 in Section 4.2) are the two main groups of obstacles that cause occlusion of objects and interruption of tracks. The case of the traffic sign is very unlikely to be improved by a new motion model since the objects that get closer to the road sign are not well detected by the sensor, as shown in Section 4.2 by Figure 4.4. The bounding box is squeezed and the position of this box is going backward. The object is perceived as if it is decelerating, as Figure 4.5 shows, and a constant acceleration model will not improve this situation because it will continue the decelerating phase. These interruptions are generated by the quality of the detections (position and bounding box) and the fact that the Intersection over Union is required to be greater than zero to perform associations between tracks and detections.

The case of the statues on the roundabout is also very difficult to improve, due to the distance of the sensor from the area of interest and the nature of the obstacles, which occupy a large part of the scenario. However, for the sake of completeness, it is better to test the algorithm with this motion model, to see the results.

The tracks obtained from the two tracking systems are very similar; there are only a few cases where they differ, and one of these is the track of a vehicle that is interrupted in the constant speed case because the vehicle is occluded for a few frames by the statues on the roundabout. The constant acceleration model manages to track it even though the track is very noisy, due to the presence of the obstacles. Figure 5.18 shows a bird's eye view of the track under analysis computed with both CA and CS motion models, with the latter case resulting in two tracks and not a unique one due to the obstacles.

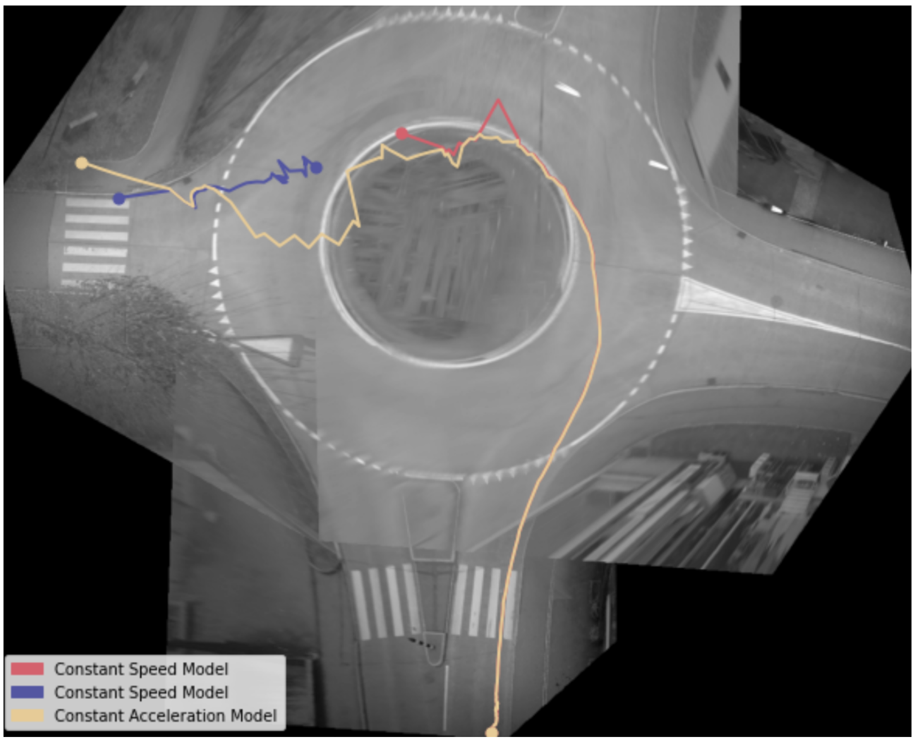


Figure 5.18: Bird's eye view of the vehicle track computed using CA and CS motion models. In both cases, the trajectory is very noisy behind the statues and towards the end.

This is the only case among the ones of trajectories that occupy the region behind the statues of the roundabout, so it is not possible to conclude that the constant acceleration motion model helps in solving these problematic cases.

Regarding general results on the tracking system, with the constant acceleration model

9 *False Positives* tracks are produced, while the number of *False Positives* tracks with the constant speed model is 11. Moreover, with the constant acceleration motion model there are 5 cases of *Identity Switches* and 3 targets that are not tracked by the tracking system, while with the constant speed model there are 4 cases of *Identity Switches* and 2 non-tracked targets. Table 5.1 shows these results for an easier comparison.

	MHT with CA model	**MHT with CS model**
False Positives	9	11
Identity Switches	5	4
Non-tracked objects	3	2

Table 5.1: Comparison of motion models

6 | Conclusion and Further Improvements

In this section, conclusions regarding the proposed algorithm will be drawn and possible improvements will be mentioned.

The proposed algorithm uses Multiple Hypothesis Tracking (MHT) to track different road users in complex urban scenarios. In particular, the algorithm adds to the classical MHT approach a constraint in the association phase and a strategy to detect fused objects based on bootstrap confidence intervals. The former consists of requiring that the Intersection over Union between the bounding box of the track and the bounding box of the detection to be associated with the track must be bigger than zero. The latter is used whenever two different tracks are associated with the same detection: two confidence intervals of the volumes of the objects targeted by the two tracks are built based on previous associations; if the volume of the new detection to be associated with the two tracks is outside both the confidence intervals, then the detection is flagged as a fused object, i.e.,it is likely that two road users have been merged into one unique detection by the sensor capturing the images.

The algorithm uses a constant speed motion model to describe the motion of the road users. Even though this motion model is too general and does not account for the motion properties of different objects, there are no problems that can be traced back to it.

The main contribution of this work is the implementation of a tracking system that uses Multiple Hypothesis Tracking in a complex urban scenario, that is, a scenario that is not characterized only by pedestrians, together with the design of the strategy to detect merged objects. By testing this algorithm on a data set containing detections coming from a video of a roundabout in Switzerland, it was possible to show that in most of the cases MHT is not necessary and the best hypothesis is the one that associates the closest detections. However, MHT played a key role in specific situations, like the cases of merged objects, as shown in Section 4.2.3. In addition to this, the strategy designed to tackle the problem of merged objects has proven successful. The algorithm is able to solve all the cases of road users fused in one detection for multiple frames: not only have these cases been correctly flagged as fused, but also the corresponding tracks remained separated and there was no case of identity switch, i.e.,the tracks did not change their targets as a consequence of a road user fused with another one in one detection.

In addition to this, the effect of kinematic constraints on the output of the tracking system has been analyzed. This type of constraints are meant to control the type of associations that are performed by the algorithm, preventing the ones that would cause speed or acceleration to have unfeasible values in an urban context. The conclusion is that these constraints make the algorithm way less flexible, reducing the capability of the tracking system to capture the noise in the data and filter it out. As such, this type of constraints should always be avoided and the model should perform the associations that fall in the gating area of the tracks, as a classical MHT system requires.

Regarding the improvements, there are a few things that can be mentioned. First of all, the strategy that uses bootstrap confidence intervals to detect fused objects works well but has been overused up until now. A tighter control should be implemented in order to guarantee that the strategy is used only when two distinct tracks, targeting two different road users, are associated with the same detection. The cases in which two tracks belonging to the same object are associated with the same detection or the cases in which the track of a stationary object (e.g., a road sign) is associated with the same detection of a real road user should not be considered as possible cases of fused detections, and thus the confidence intervals should not be computed and used. This happens in this particular dataframe and helps tracks that should be discarded remain active and enter the best global hypothesis. Once this control has been carried out, the strategy could reach higher accuracy in detecting fused objects.

The second improvement that can be mentioned is about the appearance score, which in this framework is the logarithm of the intersection over union between two bounding boxes: the one of the track and the one of the associated detection. This score is very simple and does not take into consideration the shape of the bounding box. A possibility could be to follow the strategy described in [7] and implement a classifier that uses the previous bounding boxes to understand if the bounding box of the new detection is coherent with the previous ones.

Last but not least, it is worth remembering that the proposed algorithm behaves poorly with occluded objects. Perhaps it could be possible to understand if an object is going into an area where it will be occluded by looking at the size of the bounding box. As noted in the case study discussed in Chapter 4, whenever a vehicle approaches an area hidden by the road sign, its bounding box squeezes. This situation could be spotted by comparing the size of the bounding box with the confidence interval of the volume of the track. Once the algorithm is able to recognize a case of occlusion, then it will be easier to solve this issue.

In general, it is possible to say that the algorithm performs well in tracking objects in urban scenarios and is able to overcome most of the problematic cases that come from this type of data, especially the cases of road users fused into one detection, thanks to a specific strategy implemented to tackle this problem. The Multiple Hypothesis Tracking system coupled with the strategy that uses bootstrap confidence intervals to detect cases of merged objects can help in improving the tracking performances of single hypothesis trackers and in achieving high accuracy in tracking the objects in any urban scenario.

Bibliography

[1] S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, 2004. doi: 10.1109/MAES.2004.1263228.

[2] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House radar library. Artech House, 1999. ISBN 9781580530064. URL `https://books.google.it/books?id=lTIfAQAAIAAJ`.

[3] S. Busygin. A new trust region technique for the maximum weight clique problem. *Discrete Applied Mathematics*, 154(15):2080–2096, 2006. ISSN 0166-218X. doi: https://doi.org/10.1016/j.dam.2005.04.010. URL `https://www.sciencedirect.com/science/article/pii/S0166218X06000497`. International Symposium on Combinatorial Optimization CO'02.

[4] I. Cox and S. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996. doi: 10.1109/34.481539.

[5] I. Cox and S. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996. doi: 10.1109/34.481539.

[6] R. Kaucic, A. Amitha Perera, G. Brooksby, J. Kaufhold, and A. Hoogs. A unified framework for tracking through occlusions and across sensor gaps. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 990–997 vol. 1, 2005. doi: 10.1109/CVPR.2005.53.

[7] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg. Multiple hypothesis tracking revisited. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4696–4704, 2015. doi: 10.1109/ICCV.2015.533.

[8] T. List, J. Bins, J. Vazquez, and R. Fisher. Performance evaluating the eval-

uator. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 129–136, 2005. doi: 10.1109/VSPETS.2005.1570907.

[9] A. Milan, K. Schindler, and S. Roth. Challenges of ground truth evaluation of multi-target tracking. In *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 735–742, 2013. doi: 10.1109/CVPRW.2013.111.

[10] P. M. Pardalos and J. Xue. The maximum clique problem. *Journal of global Optimization*, 4(3):301–328, 1994.

[11] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu. Multi-object tracking through simultaneous long occlusions and split-merge conditions. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 666–673, 2006. doi: 10.1109/CVPR.2006.195.

[12] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979. doi: 10.1109/TAC.1979.1102177.

[13] R. Schubert, E. Richter, and G. Wanielik. Comparison and evaluation of advanced motion models for vehicle tracking. In *2008 11th international conference on information fusion*, pages 1–6. IEEE, 2008.

[14] H. Wu, A. C. Sankaranarayanan, and R. Chellappa. Online empirical evaluation of tracking algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1443–1458, 2010. doi: 10.1109/TPAMI.2009.135.

[15] P. R. Östergård. A new algorithm for the maximum-weight clique problem. *Electronic Notes in Discrete Mathematics*, 3:153–156, 1999. ISSN 1571-0653. doi: https://doi.org/10.1016/S1571-0653(05)80045-9. URL `https://www.sciencedirect.com/science/article/pii/S1571065305800459`. 6th Twente Workshop on Graphs and Combinatorial Optimization.

A | Appendix A

Computations to prove that the *motion score* has the formula reported in 1.14. As showed in Section 1.2.2, the increment in the motion score between one frame and the following one is given by the subsequent formula:

$$\Delta S^i_{mot}(k) = \ln\left(\frac{p(y_{i_k}|y_{i_{1:k-1}}, D_{i_{1:k}} \subseteq T_i)}{p(y_{i_k}|D_{i_k} \subseteq B)}\right) \tag{A.1}$$

The assumptions on the probability distributions that appear in the equation above are:

$$p(y_{i_t}|y_{i_{1:t-1}}, D_{i_{1:t}} \subseteq T_i) = \mathcal{N}(y_{i_t}; \hat{x}^i_t, \Sigma^i_t) \qquad p(y_{i_t}|D_{i_t} \subseteq B) = \frac{1}{V}$$

where $\hat{x}^i_t$ is the position predicted by the Kalman Filter, Σ^i_t is the covariance matrix of the filter at time t and V is the measurement space, i.e.,the image area.

To begin, write in an explicit way the probability distribution $\mathcal{N}(y_{i_k}; \hat{x}^i_k, \Sigma^i_k)$:

$$p(y_{i_k}|y_{i_{1:k-1}}, D_{i_{1:k}} \subseteq T_i) = \frac{1}{\sqrt{(2\pi)^n|\Sigma^i_k|}}\exp\left(\frac{(y_{i_k} - \hat{x}^i_k)^T(\Sigma^i_k)^{-1}(y_{i_k} - \hat{x}^i_k)}{2}\right)$$

with $|\Sigma^i_k|$ being the determinant of Σ^i_k. It is worth noticing that the numerator of the exponent is the expression of the gating area of the tracks, showed in equation 1.13, which has been labeled as d^2. This notation will be adopted here too. By replacing the expressions of the probability distributions in A.1 and by using the properties of the logarithm it is possible to obtain the formula expressed in 1.14:

$$\ln\left(\frac{p(y_{i_k}|y_{i_{1:k-1}}, D_{i_{1:k}} \subseteq T_i)}{p(y_{i_k}|D_{i_k} \subseteq B)}\right) = \ln\left(\frac{V}{\sqrt{(2\pi)^n|\Sigma^i_k|}}\exp\left(\frac{d^2}{2}\right)\right) =$$

$$= \ln\left(\frac{V}{\sqrt{(2\pi)^n}}\right) + \ln\left(\frac{1}{\sqrt{|\Sigma^i_k|}}\right) + \ln\left(\exp\left(\frac{d^2}{2}\right)\right) =$$

$$= \ln\left(\frac{V}{(2\pi)^{\frac{n}{2}}}\right) - \frac{1}{2}\ln(|\Sigma^i_k|) - \frac{1}{2}d^2 = \Delta S^i_{mot}(k)$$

List of Figures

List of Tables

Ringraziamenti

Dopo sei anni sono giunto alla fine di questo lungo viaggio, che senza dubbio mi ha dato più di quanto mi aspettassi. In questi anni ho stretto amicizie che dureranno tutta la vita, imparato lezioni che mi hanno fatto crescere personalmente e professionalmente, incontrato l'amore in Svezia dove ho avuto la fortuna di vivere per un anno. Sono stati di gran lunga gli anni più felici della mia vita.

Prima di tutto voglio ringraziare Viscando per avermi dato l'opportunità di sviluppare questo progetto. Lavorare con tutti voi è stato un grande piacere. In particolare, voglio ringraziare Yury per essere stato un grande supervisore: i tuoi consigli sono stati molto preziosi e il tuo aiuto è stato molto importante per me.

Voglio anche ringraziare il mio supervisore qui al Politecnico, il professor Simone Vantini. Dal primo giorno lei è stato molto comprensivo e disponibile ad aiutarmi con grande umanità. La mia situazione non era semplice ma la ringrazio per la grande disponibilità dimostratami.

Voglio ringraziare mio fratello, mia mamma e mio papà che mi hanno accompagnato lungo questo percorso sostenendomi in mille modi diversi. Siete la mia forza. Il vostro affetto e amore mi hanno aiutato a farcela e a superare ogni difficoltà. La vostra pazienza verso il mio essere scorbutico durante le sessioni è stata encomiabile. Vi sarò per sempre grato per tutte le esperienze che mi avete permesso di vivere. Se sono qui oggi è soprattutto merito vostro.

Voglio ringraziare i miei amici di "Teosomelier" (rigorosamente con una m per limite al numero di caratteri su whatsapp): siete il miglior gruppo di amici che l'università potesse farmi conoscere. Mi mancheranno le nostre partite a "King" e non mi mancheranno le sessioni di studio, che però sono state fondamentali.

Voglio ringraziare i miei amici. Dalle scuole elementari fino all'università ho incontrato persone fantastiche che porto sempre nel mio cuore in qualsiasi paese mi trovi.

Ringrazio i miei amici del liceo e di Cologno, che porto sempre nel mio cuore in ogni paese in cui mi trovi. Anche se non riusciamo a vederci più come prima, sappiate che siete molto importanti e speciali per me.

Ed ora voglio ringraziare Leanne, la mia compagna di vita, la persona che più di tutti mi è stata vicino in questi due anni e la cui importanza è inestimabile. Ti ringrazio per essere la splendida persona che sei, per avere sempre le parole giuste per consolarmi nei momenti difficili e per avermi reso una persona migliore. Il tuo supporto è stato essenziale per me. Fianco a fianco abbiamo fatto tanta strada, e chissà quanta ne faremo ancora.

Grazie a tutti dal profondo del mio cuore.

Acknowledgements

After six years I have come to the end of this long journey, which has unquestionably given me more than I expected. In these years I have made friendships that will last a lifetime, learned lessons that have made me grow personally and professionally and met love in Sweden, where I was lucky enough to live for a year. They were by far the happiest years of my life.

First of all, I want to thank Viscando for giving me the opportunity to develop this work. Working with you all was a great pleasure. In particular, I want to thank Yury for being a great supervisor. Your advice was very precious and your help has been very important for me.

I also want to thank my supervisor here at Politecnico, Professor Simone Vantini. From the first day you have been very understanding and willing to help me with great humanity. My situation was not simple but I thank you for the great availability shown to me.

I want to thank my brother, my mom and my dad, who have been with me along this path, supporting me in a thousand different ways. You are my strength. Your affection and love helped me make it through and overcome every difficulty. I will always be grateful to you for all the experiences you have allowed me to have. If I'm here today, it's mainly thanks to you.

I want to thank my friends of "Teosomelier" (with only one "m" due to the limit in number of characters on Whatsapp): you are the best group of friends that the university could lead me to. I will miss our games at "King" or "Briscola" and for sure I will not miss the study sessions, even though they were fundamental.

I thank my friends from high school and from Cologno, whom I always carry in my heart wherever I am. Even if we can't see each other like before, you should know that you are very important and special to me.

And now I want to thank Leanne, my life partner, the person who has been closest to me in these two years and whose importance is inestimable. Thank you for being the wonderful person you are, for always having the right words to console me in difficult times and for making me a better person. Your support has been essential for me. Side by side we have come a long way, and who knows what beautiful experiences the future holds for us.

Thank you all from the bottom of my heart.