UCLouvain

epl

École polytechnique de Louvain

Fault-tolerant crossroad multi-tracking in urban areas

Author: **Pierre CAPRASSE**
Supervisor: **Benoît MACQ**
Readers: **Dani MANJAH, Benoît GERIN, Antoine VANDERSCHUEREN**
Academic year 2021–2022
Master [120] in Mathematical Engineering

Acknowledgments

This master thesis has been written to fulfill the graduation requirements of the mathematical engineering program at UCLouvain.

I would like to express my gratitude to my thesis supervisor Professor Benoît Macq, who offered precious advice and guidance. Moreover, I am extremely thankful to my supervisor Dani Manjah for his constant support, feedback and assistance. The present form of this thesis could not have been achieved without his help.

I would also like to thank Benoît Gerin and Victorien Sonneville for their valuable advice. Finally, I am grateful to Antoine Vanderschueren for agreeing to be a member of the jury.

I sincerely hope that you will enjoy reading this thesis and that it offers useful insights into fault-tolerant multi-camera tracking.

Pierre Caprasse

Kraainem, June 2022

Abstract

Due to the expansion of the number of smart cities, city-scale multi-camera vehicle tracking is gaining a lot of interest. Several robust systems have already been implemented in an attempt to address this demand. However, heavy occlusion and faulty detections can greatly reduce the performance of these systems. In this thesis, a simple multi-camera vehicle tracking is implemented with an additional Feedback Reconstruction Algorithm, called interchangeably FRA or feedback. It is a new principle that reconstructs missing detections due to faulty cameras at a crossroad. The algorithm is based on detections made by other cameras. It also uses additional information collected at the previous time frame. The reconstruction's purpose is to avoid losing a car's track. The system has been tested in several scenarios where detections have been deliberately removed. The evaluation has been performed on two crossroads using a dataset provided by the AI city challenge [15]. Considering a 50 to 60% missing detection rate on a camera, the system can recover up to 40% of the initial HOTA score (without missing detection) of the faulty camera, and 37.5% of the global HOTA score of the intersection. These results, obtained by simulation, demonstrate the performance and effectiveness of the proposed recovering detection algorithm.

Keywords: Object detection, object tracking, multi-camera network, fault-tolerant.

Nomenclature

Acronym	Description
AI	Artificial Intelligence
CNN	Convolutional Neural Network
FPS	Frames Per Second
FRA	Feedback Reconstruction Algorithm
GPS	GLobal Positioning System
HOTA	Higher Order Tracking Accuracy
ICT	Inter-Camera Tracking
IOU	Intersection-Over-Union
MOMCT	Multi-Object Multi-Camera Tracking
MTMCT	Multi-Target Multi-Camera Tracking
SCT	Single Camera Tracking
SORT	Simple Online and Realtime Tracking
TNT	TrackletNet Tracker
UAV	Unmanned Aerial Vehicles

Contents

Chapter 1

Introduction

Currently, half of the global population lives in cities while they only represent 2 % of the earth's surface. This concentration is expected to reach two-thirds of the population by 2050 [19].The improvement of urban services' efficiency is therefore essential. One of the proposed solutions is to transform our towns into smart cities. Smart cities integrate an ecosystem of connected objects able to offer digital services to all major urban activities: transport, health, energy, political and economic life.

The analysis of large-scale traffic flows is a key consideration for smart cities. It enables real-time decision-making, which dramatically speed up reaction time and hence the security of a town. The interest in city-scale traffic management using multi-object multi-camera tracking (MOMCT) has therefore grown exponentially in recent years. Several MOMCT systems have been implemented dealing among others with challenges such as similar vehicle models during identification, occlusion, variation in view angle, etc. Most of them follow a two-step approach: single camera tracking (SCT) and inter-camera tracking (ICT). SCT consists of tracking cars on each camera individually. Algorithms such as deepSORT and trackletNet tracker are often used. ICT aims to associate vehicle trajectories across multiple cameras. It uses information such as appearance features, and spatial and temporal information. The association of trajectories from the same car is carried out using algorithms such as hierarchical clustering.

The systems referenced in the literature are already quite effective. However, the performance of MOMCT in terms of tracking accuracy can be severely degraded when faults occur. As explained in article [10]: "*Faults may be caused by unpredictable software errors (e.g., in the image processing, feature extraction and data association modules), hardware malfunctions (e.g., in the camera mechanical parts or lens), or as the result of a malicious attack* ". In order to manage traffic

in a city like Brussels, thousands or even tens of thousands of cameras are needed. Errors in the process are therefore inevitable.

Obviously, research has already been conducted on this subject. The principle is predominantly based on redundant information and carries the errors through a voting system. Found errors are generally filtered out. Some of the proposed solutions are based on a *certainty map* combined with a threshold [7]. In the specific case of an orientation error, the article [7] proposes a solution to correct the orientation. However, few other solutions aim at correcting an error instead of simply deleting it.

This master thesis would like to suggest a new solution to a specific set of errors due to missing detections at crossroads. The causes of this types of errors can vary. For example, errors may be due to heavy occlusions, a software error, or inaccuracies provided by the detection algorithm. It will be shown that reconstructing these detections will significantly improve the accuracy of vehicle tracking. The reconstructing approach involves the implementation of a simple MOMCT algorithm composed of 3 main steps: single camera tracking, clustering and matching. The aim of this thesis is not to compete with other existing MOMCT systems, but rather to show the positive impact of reconstructing missing detections. The principle used is interchangeably referred to as Feedback Reconstruction Algorithm (FRA) or feedback. It is based on the following rationale: if two cameras detect the same car at a certain time frame, it should also be the case at the next time frame. If this is not the case, the missing detections will be reconstructed, unless the object is moving out of the field of view of one of the cameras. The position and shape of the missing detections are estimated with the previous time information and the detection of the other camera. Given the operating process of the feedback, it is ineffective when one of the cameras is not working at all but is of great benefit to compensate for partial detection issues by one of the cameras.

The main contribution of this master thesis is the feedback process, a new principle to reconstruct missing detections at an intersection.

The content of this thesis is organized as follows. Related works are presented in chapter 2. In chapter 3, the feedback and the implemented MOMCT algorithm are explained. Then, the performance of the feedback is assessed in chapter 4. In chapter 5, some future perspectives are described. Finally, chapter 6 draws the conclusion of the thesis.

Chapter 2

State Of the Art

The analysis of the state of the art is divided into two main parts. The first part explains the methods currently used for multi-object multi-camera tracking. Given the great attention given to smart cities in recent years, this is a very popular research topic with already quite advanced solutions. The purpose of this thesis is not to compete with these solutions, but rather to complement them by addressing a known issue in the field of MOMCT due to detection errors coming from faulty cameras. The second part of this chapter is therefore dedicated to articles focusing on the design of fault-tolerant algorithms.

2.1 Multi-object multi-camera tracking

MOMCT is a challenging topic. Several kinds of problems can occur during the tracking of multiple objects. Indeed, unreliable vehicle detection, similar vehicle models, heavy occlusions and variations in view angle are all fundamental issues that need to be solved.

There are many ways to deal with these issues. However, there is a common approach used in the literature, which has also been applied to this thesis. This general approach will first be described. Afterwards, some specific methods will be summarised.

2.1.1 Common methodology

Most MOMCT systems follow a two-step approach as shown in figure 2.2. Single Camera Tracking (SCT), being the first step, consists of tracking cars on each camera individually. The result of SCT is a tracker for each detected object. It contains information about the tracked object, such as its position.

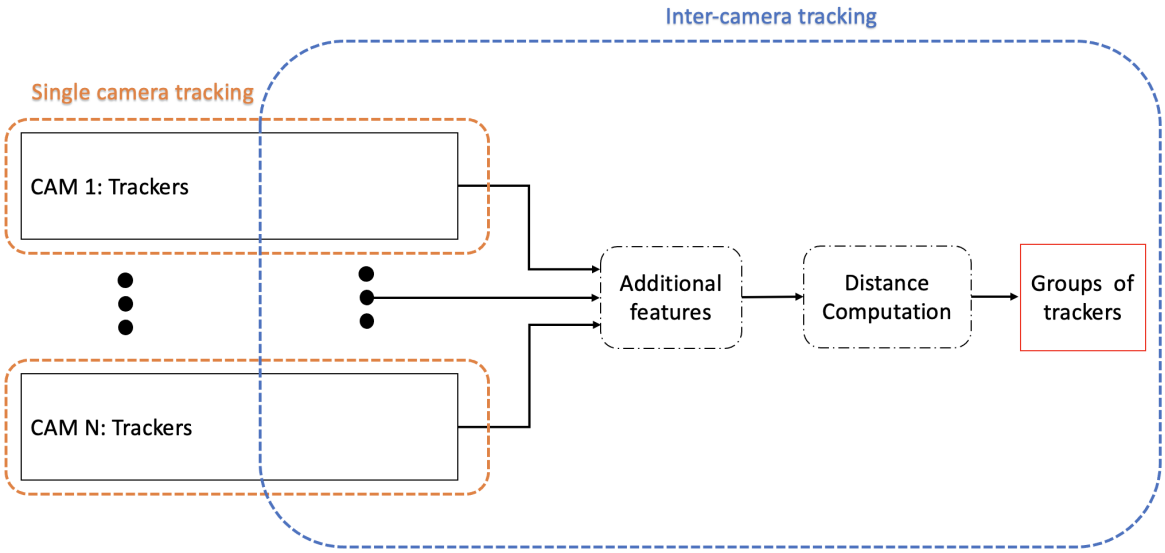


Figure 2.1: MOMCT systems two-step approach: single camera tracking and inter-camera tracking.

The second step is Inter-Camera Tracking (ICT), which aims to associate the vehicle trackers across multiple cameras. Many different methods have already been tested. However, common characteristics can be observed. First, additional features are collected and added to the trackers. This information is used to distinguish different cars, for example by identifying their brand. Then, distances are computed between the trackers based on their collected features. Of course, the distance are computed in such a way that two trackers coming from the same car must be significantly smaller than two trackers coming from different cars. Each feature can be compared resulting in a distance. The different distances have to be combined in order to have only one measure to compare trackers with each other. Finally, with the found distances, trackers coming from the same car can be associated with each other.

The single camera tracking step, the additional features, the trackers' distances computation and the trackers association method vary from article to article. Several options chosen in the literature will be presented next.

2.1.2 Single camera tracking (SCT)

Single camera tracking aims to identify objects (eg, cars, pedestrians, etc) and track them across multiple frames. Multiple types of SCTs exist, but the tracking-by-detection scheme has proven to be efficient. It consists of two parts. First, the objects have to be detected on each image, then they can be linked together in order to track the different objects.

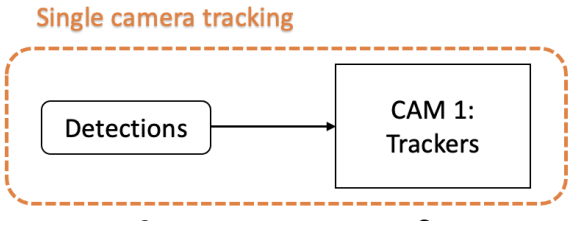


Figure 2.2: Tracking-by-detection SCT scheme: detection step and tracking algorithm.

Five state-of-the-art detection algorithms were assessed in [18]. They are YOLOv4, mask R-CNN, cascade RCNN, centerNet, and efficientDet. The latter showed to be the best performing one, even though each method has its pros and cons.

For the second step, several tracking methods exist in the literature. Some are simple, like SORT, which uses a kalman filter in combination with a Hungarian algorithm. Others, like deepSORT, are more advanced and incorporate visual information extracted by a Convolutional Neural Network (CNN) [9],[18], [3]. Another commonly used method is the trackletNet tracker (TNT) [5].

For the first part of this thesis, a simple MOMCT is implemented. Therefore, the SORT algorithm has been chosen and is detailed in the methodology.

2.1.3 Features and restrictions

Three main categories of features are generally collected: appearance, spatial and temporal features. Spatial and temporal features are often combined together. Often, some features are already used in the tracking step of the SCT and need therefore not to be computed anymore.

Appearance features

Two sorts of appearance features can be extracted from a tracked object. It could be typical metadata such as color [18] [5], type [18] [5] and brand [5]. Or it could be features generated by a convolutional neural network (CNN) [9] [18]. The CNN is a deep learning approach that extracts features from an image.

Spatial information

A car viewed by different cameras at the same time produces trackers in each of these cameras. With homography transformation, it is possible to extract the GPS coordinates from the position of the tracker. If GPS coordinates were always accurately estimated, this feature alone would be sufficient to match the trackers of the same car together and to create a relatively good working MOMCT when dealing with cameras having an overlapping field of view. It will therefore be the only feature used in the methodology of this thesis. However, estimate errors occur during the GPS localisation estimation process. In order to get a perfect match in any situation, additional features need to be collected. For example, the vehicle's orientation is sometimes used to complement GPS coordinates. [18].

Temporal information

An object cannot appear two times on the same frame of a camera simultaneously. Therefore, tracks coming from the same cameras cannot be matched together. This basic idea narrows the matching candidates for the track association.

Temporal and spatial information combined

Temporal and spatial information can be combined in order to model the movement of the objects. It can be a simple linear prediction model [9] or more complex models that incorporate traffic rules. Article [18] aims to model traffic rules with pre-defined zones in the image, while article [5] aims at constructing them automatically. The prediction of the movement of an object is essential when cameras have a non-overlapping field of view.

2.1.4 Distance Computation

A distance can be computed between each feature of different trackers. A weighted aggregation method is often used to compute the distance [9], [18]. It consists of a weighted sum of all computed distances. For example, when four features are collected: color, type, ID and orientation, a distance is computed for each of them: D_{color}, D_{type}, $D_{vehicle}$ and $D_{orientation}$. A distance metric can then be computed by combining the distances as follows:

$$D = D_{vehicle} + \alpha D_{color} + \beta D_{type} - \gamma D_{orientation} \tag{2.1}$$

where *alpha*, *beta* and *gamma* are the weight parameters. [18].

Note that it is desired that two trackers from the same object are significantly closer than two trackers from different objects. The distances are often put into a matrix in order to perform track association optimally. Weighted aggregation is not the only way to combine the features. For example, article [5] uses a Siamese TrackletNet.

In this thesis, only one spatial feature is used, the object's position. This step is therefore unnecessary.

2.1.5 Track association

This last task is to form groups of trackers coming from the same object using the distance matrix computed in the previous step. The most used method is hierarchical clustering [9], [5]. It is therefore also implemented for this thesis.

One of the concerns in track association is to define an accurate threshold to determine whether two tracklets are matching or not. A solution proposed by [18] is to use an uncertainty-based tracklets matching method.

2.2 Fault detections

When the camera sensors are working properly, the various algorithms described above are very robust. However, when a network of cameras is made up of hundreds of them, one or more will inevitably produce errors. This can greatly reduce the performance of the tracking. The errors must therefore be spotted and corrected. Several solutions have been designed for this problem and are presented in this section.

2.2.1 Multi-UAV error detections [17]

The issue of fault detection has already been analysed in the context of Unmanned Aerial Vehicles (UAVs). Here, cameras are placed on each UAV. The objective is to estimate the actual position of one UAV when it is detected by all the others. The tracked UAV has itself an estimate of its 3D position. All detections are compared to this estimate and the differences are then computed. Excessive differences point to inconsistent measurements or errors. In other words, the method exploits redundant information provided by several measurements of the same data to detect errors. The method used differ depending on the number observations collected.

In the context of car tracking, there are no available estimates of the car's position. Hence, the principle used to UAVs cannot be used in our case. Nevertheless, the ideas to exploit redundant information and to develop a model based on the number of observations can also be applied to identify crossroad MOMCT camera errors.

2.2.2 Orientation correction in visual sensor networks [7]

Article [7] proposes a detection algorithm that tolerates some errors based on a *certainty map*. It's a map describing the probability of having an object at a certain coordinate. For example, image 2.3 represents 5 certainty maps containing two objects.

Thanks to the use of multiple cameras, redundant information is produced that allows for tolerating faults in sensors. A voting mechanism is set up in order to detect potential sensor faults. Each camera has an equal voting impact on the result. The more votes there are on a coordinate, the greater the probability of an object being present. A threshold has to be applied to know whether or not an object is detected. A threshold of x means that if x cameras do not observe the existence of an object at a certain location in the certainty map, it will be determined as

being nonexistent. Thus, a threshold of one means that no tolerance of failure is accepted. Indeed, if one camera fails to detect the object, it will be chosen to be nonexistent. Different cases of thresholds are visible in figure 2.3. As shown in the figure, higher thresholds lead to wider detection areas of an object. The conclusion is that by using this approach and choosing an appropriate threshold, errors are tolerated but not corrected.

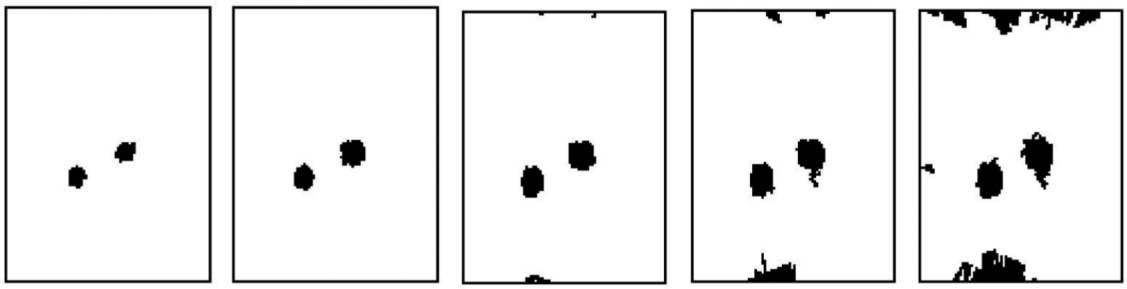


Figure 2.3: From left to right a certainty map using voting with the threshold 1, 2, 3, 4, and 5, respectively [7]

It is also possible to correct a specific type of errors coming from a faulty calibration. Once again, thanks to the redundant information, inaccuracies in a camera's orientation can be detected. With an adapted algorithm, it is then possible to correct the orientation [7].

2.2.3 Decentralised method to detect faulty cameras [10]

In article [10], a decentralised solution to detect faulty cameras in the context of camera sensor networks has been implemented. It can be described in two main steps. The first step is a protocol to elect a leader camera. In the second step, the leading camera maintains a voting matrix of its neighbors. Thanks to the latter, it decides whether or not a camera is healthy. If the camera is considered faulty, the information provided by that camera are removed.

2.2.4 Conclusion

In the literature, the issue of malfunctioning cameras has already been studied. Solutions are always based on redundant information coming from different measurements. Thanks to a voting system, errors are filtered out. In the case of orientation errors, errors are also corrected.

This thesis would like to propose a new solution to another specific case of errors: missing detections. The objective is not try and eliminate information from faulty cameras. Rather, our goal is to reconstruct missing detections thanks to redundant information produced by other cameras. Missing detections can be due to heavy occlusions, a software error or simply inaccuracies provided by the detection algorithm.

Chapter 3

Methodology

Within the context of large-scale traffic flows, MOMCT has drawn a lot of attention. A commonly used approach consisting of SCT and ICT has been described in the state of the art. The methodology used in this thesis is therefore also based on a combination of SCT and ICT. Moreover, ICT has been divided into two sub-methods: clustering and matching. Finally, an additional step, FRA, has been implemented in order to reconstruct missing detections. As a result, the methodology includes 4 main steps,as shown in figure 3.1.

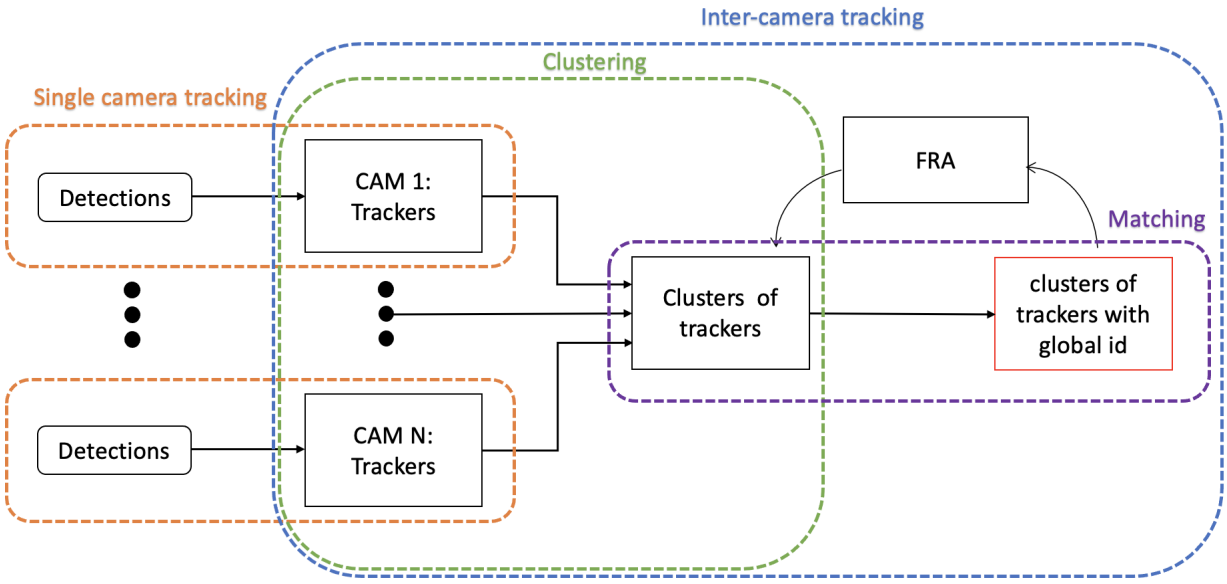


Figure 3.1: The 4 main steps used in the methodology: single camera tracking, clustering, matching and FRA.

The first step is SCT using the tracking-by-detection scheme. It aims to track the vehicle trajectories for a single camera. The goal is to track cars on each camera individually using the detections found on the incoming images. For the tracking part, the SORT algorithm is used. This first step is described in detail in the first section of this chapter.

Subsequently, ICT is used and aims to associate the vehicle trackers across cameras. In this methodology, it has been divided into two sub-methods: clustering (step 2) and matching (step 3). The clustering projects each tracker on an *occupancy map* and groups the closest trackers together. These groups are called clusters. A cluster is supposed to contain trackers belonging to the same car. The matching will then associate the clusters in time. Clustering and matching are described in sections 3.2 and 3.3 respectively.

Single camera tracking, clustering and matching form together an algorithm of MOMCT. In section 3.4, this system is tested on a dataset provided by AI city challenge [15]. The results are not as robust as those proposed in the state of the art. Nevertheless, the objective of this thesis is not to try to compete with other research but rather to improve the system when some errors are detected. For this purpose, an additional feedback step , FRA, has been added to the algorithm (step 4). The goal of FRA is to reconstruct missing vehicle detection data. It is presented in section 3.5.

To evaluate our system, the HOTA is chosen and described in the last section of this chapter.

Summing up, the methodology used consists of single camera tracking, clustering and matching. Besides, an additional step has been implemented to correct missing detections. To evaluate the system experimentally, the HOTA metric is used.

3.1 Single Camera tracking

The first step is applied using a single camera, independently of the others. Each camera sensor is comprised of a detector and a tracker. The detector processes the raw signal to reveal the presence of objects of interest (e.g. cars, pedestrians, etc.). It delivers a list of bounding boxes which is an input for the tracker. [1]

The role of the tracker is to associate detections coming from the same object in time. For example, if for a certain number of frames, 3 cars are detected, it is necessary to track them across the subsequent frames and to allocate a local id for each of them. This is a well-known problem referred to as multiple object tracking. A common method used to address this issue is SORT, which stands for "Simple Online and Real Time Tracking". It achieves an accuracy level comparable to state-of-the-art online trackers [2]. The SORT algorithm is a combination of the Kalman filter and the Hungarian algorithm. The understanding of the Kalman filter is therefore necessary and is presented in the next section. The understanding of the Hungarian algorithm is less important but can be found in article [8]. Afterwards, SORT will be explained in more detail.

3.1.1 Kalman Filter

The kalman Filter is an estimation algorithm combining a prediction from a system model and a measurement in order to give a more accurate prediction.

The Kalman filter is used to estimate the position of an object x_k (e.g. cars. pedestrians, etc.). An external measurement y_k of the position is always available. The issue is that this measurement is often not sufficiently accurate. It is represented by an observation noise v_k. It is assumed to be a zero-mean Gaussian white noise. Thus, the measurement can be represented by equation 3.1.

$$y_k = Cx_k + v_k \tag{3.1}$$

Note that when y_k measures the position, C is simply the identity matrix. In general, y_k does not have to measure it directly.

[1] A bounding box is an imaginary rectangle surrounding an object aiming to describe its spatial location.

The idea of the Kalman filter is that the modelling of an object's dynamics can generate an estimate of the object's position. Combining this estimate with the measurement y_k could give us a more accurate estimate. This idea is represented in figure 3.2. The initial state estimate is $\hat{x}_{k-1}$ and the available measurement of the new position is y_k. If a model of the system is available, it would provide a prediction of the state estimate $\hat{x}_k^-$. The Kalman filter would then compute an optimal state estimate $\hat{x}_k$ from $\hat{x}_k^-$ and y_k. The $^-$ in $\hat{x}_k^-$ is there to highlight the difference between the estimate made with the system model $\hat{x}_k^-$ and the one made from the combination of $\hat{x}_k^-$ and the measurement y_k which gives $\hat{x}_k$ without $^-$.

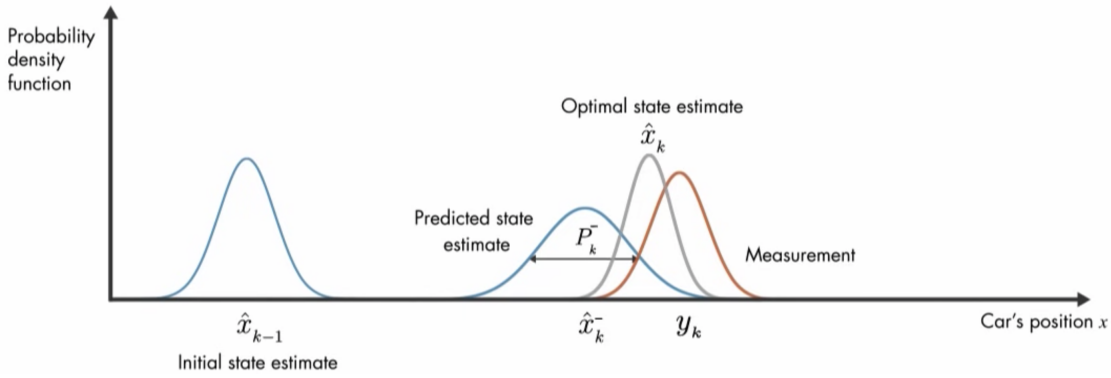


Figure 3.2: The principle of the Kalman filter: with a system model, a predicted state estimate can be computed from an initial state. Afterwards, an optimal state estimate can be computed by combining the predicted state estimate and an external noisy measurement [14].

Thus, equations describing the car dynamics are required. The Kalman filter is based on a linear model. Therefore, the motion of the car can be described by the following equation:

$$x_k = Ax_{k-1} + Bu_k + w_k \tag{3.2}$$

where x_k represents the state vector. In this example, x_k represents the car's position. The control vector, u_k, would be the velocity of the car. $Ax_{k-1} + Bu_k$ is the equation that models the linear prediction, and w_k is the process noise. It is necessary to represent the uncertainty about the car's position. For example, this would take into account the fact that the car could have run over a pothole. When using a kalman filter, the process noise is assumed to be a zero-mean Gaussian white noise.

Knowing the car dynamics 3.2, we could model the system with equation 3.3. It gives a first prediction of the new position of the car $\hat{x}_k^-$. As explained, there is uncertainty about the position of the car x_{k-1}. So there is a covariance matrix P_{k-1} associated with it. It has also to be updated with the new prediction. This is done with equation 3.4.

Prediction Equations:

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_k \tag{3.3}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{3.4}$$

The prediction step is followed by an update step. It combines the prediction $\hat{x}_k^-$ with the measurement y_k in order to find a more accurate estimate of the position $\hat{x}^k$. It is done by equation 3.6. We can observe that the new value $\hat{x}_k$ is equal to the previous value $\hat{x}_k^-$ where we add the difference between the prediction and the measurement multiplied by some weight K_k. This weight is called the Kalman gain and is calculated such that it minimizes the a posteriori error covariance. The exact value is calculated with equation 3.5. Finally, we have to update once again the covariance matrix (3.7. Notice that to estimate the current state, only the measurement, the estimated state and the covariance matrix from the previous time step are needed. All other past information is not needed.

Update Equations:

$$K_k = \frac{P_k^- C^T}{CP_k^- C^T + K} \tag{3.5}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-) \tag{3.6}$$

$$P_k = (I - K_k C)P_k^- \tag{3.7}$$

In summary, using a linear prediction model gives a way to make a first prediction of the car's position. This is called the prediction step and can be done with the equations 3.3 and 3.4. Then we have an external way that gives us an approximation of the position. Finally, the kalman filter will compute a more accurate estimate from the prediction and the measurement, using equations 3.5,3.6 and 3.7.

3.1.2 SORT algorithm

The method tracks multiple objects across frames using the Kalman filter. At each time frame, the objective is to associate the new detections with some tracks that were made from earlier detections. For this purpose, the Kalman filter will be a useful tool. This method is described in 4 parts. First, the detection algorithm is described. Secondly, the link with the Kalman filter is explained in the estimation model part. Thirdly, the association of detections to tracks is discussed. Finally,the update, creation and deletion of tracked objects is presented.

Detection

Each incoming image is stratified by a detector method. It delivers a list of bounding boxes which is then used as input for the tracker. Multiple efficient detectors exist nowadays. The AI city challenge provides some bounding boxes from different baselines such as *YOLOv3*, *ssd512*, and *mask-RCNN*. However, since these detections contain some inaccuracies, we opted to take the ground truth detections as input for the model. To ensure that this choice does not distort the results, an analysis is made in appendix C using the *YOLOv3* detections.

Detections from baselines and from the ground truth are both bounding boxes of cars at each time frame. The ground truth detections are supposed to be more precise since they are the handwritten solutions while the others are the output of an algorithm. In addition, the ground truth notations also contain the ids, which are identical when the detections correspond to the same car. These ids will obviously not be used during the algorithm but only to compare them with the final results.

Moreover, it should be noted that the ground truth contains the detections of the cars only when they are visible on at least two cameras at the same time, whereas the baseline detections include all detections visible on each camera, independently of each other [15].

Estimation Model

At each new time frame, new detections are made. These detections have to be linked to the correct tracker that was created thanks to detections from previous frames. To compare tracks to the new detections, it is necessary to have an estimate of the actual position of the car. As explained in subsection 3.1.1, it is possible to do so using a Kalman filter. So each track will make an estimate of the new car's position using a linear constant velocity model.

The code used for the SORT algorithm comes from article [4], as does the modeling part. As mentioned in article [4]: "The state vector of each target is modeled as $x = [u, v, s, r, \dot{u}, \dot{s}, \dot{s}]^T$, where u and v represent the horizontal and vertical pixel location of the center of the target, while the scale s and r represent the scale (area) and the aspect ratio of the target's bounding box respectively. Note that the aspect ratio is considered to be constant. "

Data Association

The estimation model predictions will now be compared to the detections. Each comparison is evaluated by an intersection-over-union (IOU) score. It is simply the area of overlap divided by the area of the union as shown in figure 3.22. These scores could be seen as the weight of a bipartite graph where one set represents the detections and the other one the predictions. Here, the Hungarian algorithm can be used to optimally associate detections and predictions. More information about the Hungarian algorithm can be found on [8]. Furthermore, it is necessary to add a minimal IOU score that rejects some associations in order to avoid false ones. This generates unmatched trackers and unmatched detections. The minimum IOU threshold is set at 0.3 which was chosen by the authors of [2]. Upon reflection, it seems to be an acceptable threshold as it amounts to about 30% of the common area between the two bounding boxes.

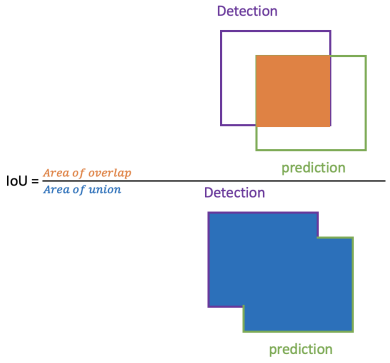


Figure 3.3: Intersection-over-union score is the area of overlap divided by the area of union between prediction and detection

Update, creation and deletion of tracked objects

The data association leads to 3 different cases. Case 1, detections are associated with trackers. Thus we use the detection as a external measurement in order to update the Kalman filter with the update equations 3.5, 3.6 and 3.7. Case 2 corresponds to an unmatched detection. In this case, a new track is created using the geometry of the bounding box with the velocity set to zero. Lastly, when a track is unmatched, a counter is activated. If this happens T_{lost} times in a row, the tracker is removed. T_{Lost} is set to 1 because it seems to be the most optimal solution according to [2].

3.2 Hierarchical clustering

As explained previously, the first stage of the general algorithm is the single camera tracking using the SORT algorithm. For instance, if 3 cars are visible at a crossroad, it will have 3 trackers on one camera and 3 other trackers on a second camera. Each tracker has its own id. The next challenge, represented on image 3.4, is to create groups of trackers from the same object across cameras. The same id could be given to each tracker belonging to the same group. The result would be a global id for trackers coming from the same car. A hierarchical clustering algorithm is used to form the groups, also called clusters. Because this method is based on distances, it requires to find a way to compute the distance between bounding boxes of different cameras. This will be done in the next section. Afterwards, the main steps of the clustering will be presented.

3.2.1 Homography distance

Since the pixel coordinates of a bounding box from one camera are not comparable with those of another, it is necessary to find a way to compare them in a common reference plane. This will allow us to compute the distance of two bounding boxes from different cameras. One way to do so is to project the middle point of the bounding box on the ground, like in figure 3.5. This is possible thanks to the homographic matrix provided for each camera in the AI city dataset [15].

The provided 3x3 homography matrix H relates the transformation between the plane of the camera and the ground plane with GPS locations. For example, point $a = (x, y)$ is a pixel coordinate on a camera that corresponds to the GPS location point $b = (x', y')$. Then it is possible to find point b from point a with:

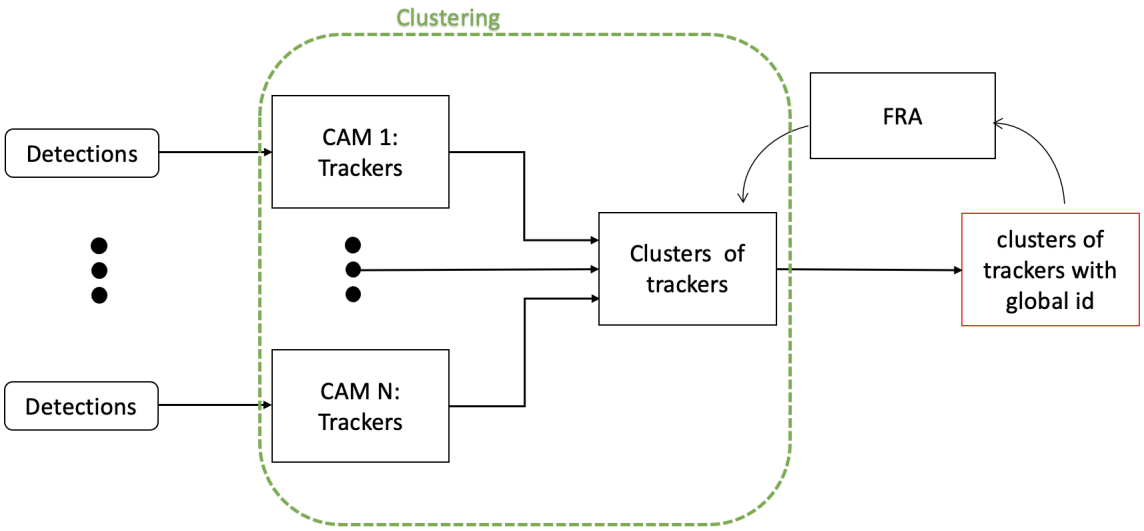


Figure 3.4: Step 2 of the methodology: clustering. It aims to create groups of trackers from the same object across cameras.

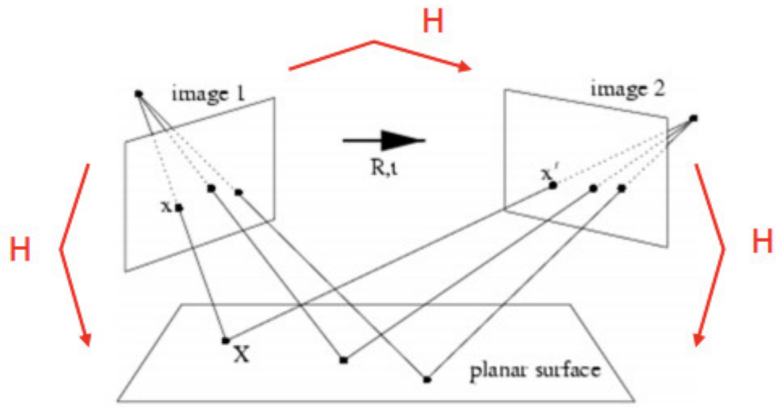


Figure 3.5: Homography projection from 2 planes on the ground plane [16].

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.8}$$

Notice that a variable s appears in the equation. This reflects the fact that the position is estimated up to a scale.

Knowing the principle of this projection, it is then simple to calculate the distance between two detections. First, project the centers of the bounding boxes with the homographic matrix and then take the euclidean distance between the two points. Note that the GPS positions are not coordinates on a flat plane. Indeed, GPS locations are angular values. However, since the distance between cars on a crossroad is very small compared to the perimeter of the earth, they can still be safely viewed as a linear coordinate system.

3.2.2 Main algorithm

Now that the notion of distance is defined, the principle of hierarchical clustering can be elaborated. To initialise the algorithm, every track is identified as one cluster. Then, the two clusters with the smallest distance will be merged together. This distance corresponds to the distance between the last detection added to each track and is computed as explained previously. This principle is repeated until the smallest distance is greater than a threshold or if all tracks are matched. The distance threshold is chosen to be 0.00016 which corresponds approximately to 1m. Knowing the geometrical sizes of a car, if the center of 2 cars is separated by a maximum distance of 1 meter, there is a high probability that it is the same radius. Otherwise, it would mean that they have collided. Of course, due to projection errors, the clustering matches sometimes different cars close to each other, even when using the threshold.

Even though the principle is quite simple, the implementation is more complex. Each time two clusters are matched, the corresponding tracks receive the same id. For instance, detection 1, coming from camera 1, and detection 2, coming from camera 2 are matched together. Thus, no more detections from camera 1 can be matched with detection 2. This would mean that the same car appears two times on the same camera at the same moment, which is not possible. In a similar way, no more detections from camera 2 can be matched with detection 1. As a result, all these possible matches have to be deleted from the remaining matching possibilities. Additionally, every distance from the new cluster to the other cluster has to be recalculated. The simplest way is to take the average from all possible distances of an element of a cluster with an element of the other cluster. For example, if a cluster is composed of two cars A and B, and another cluster of one car C, the resulting distance will be the average distance between A-C and B-C.

In order to better understand the algorithm, an example will be described with the help of the figure 3.6. As visible on the first image of the clustering step, all projections are one cluster. Then the two clusters with the smallest distance, grey

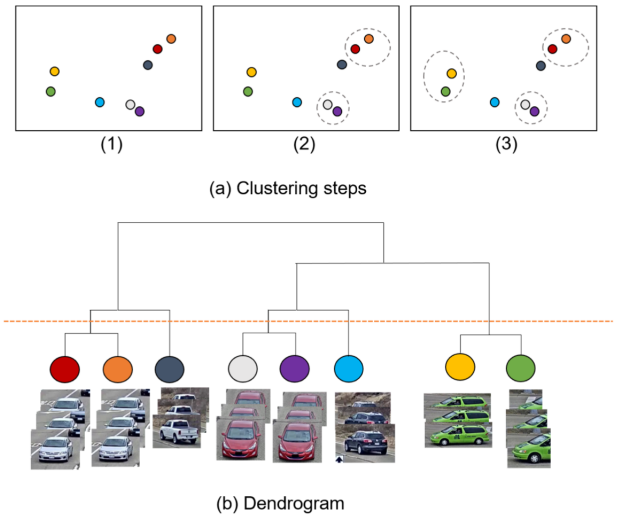


Figure 3.6: The procedure of hierarchical clustering: (a) the clustering steps, (b) the dendrogram of clusters with different threshold levels. Image provided by [5]

and purple, are matched together. As we can see on the dendrogram, these two projections correspond to the same car. Thereafter, the gray and orange clusters are matched, followed by the green and yellow ones. The smallest distance is now greater than the chosen threshold. Therefore, the algorithm stops and no more matches are made. Finally, 5 clusters are remaining, corresponding to 5 different cars.

For information purposes, if no threshold were applied, the algorithm would end with one big cluster including every projection in it. A dendrogram could then be made where each node corresponds to a match and each leaf to a projection. This is why it is called hierarchical clustering. In addition, the red line visible on the dendrogram corresponds to the threshold used.

Since the ground truth of the AI city challenge only contains cars that are visible by minimum 2 cameras, the final step will only select the clusters with a size greater than 1.

To sum up, when cameras have overlapping areas, a car can be viewed by multiple cameras at the same time. Nevertheless, the single camera tracker does not connect these tracks although they actually belong to the same car. Hierarchical clustering brings a solution to this problem by matching tracks to each other.

3.3 Matching

Now that several tracks have been grouped together thanks to clustering, we need to give them an ID that is consistent over time. This is the purpose of the matching process, shown in image 3.7. By default, each cluster receives a new non-existent id which implies that each selected cluster is considered as a new car.

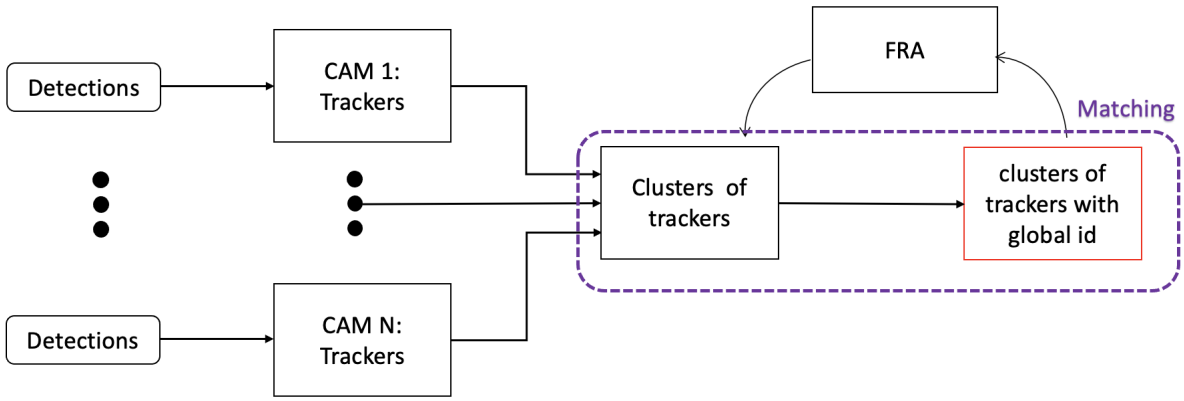


Figure 3.7: Step 3 of the methodology: matching. It aims to give the groups of trackers a consistent ID over time.

Then, these clusters are compared to the ones found in the previous time step. Remember that each selected group contains at least two tracks. A choice was made to give the same global id to all tracker as the one from a previous cluster if it had two trackers in common. Which is usually the case.

It can sometimes happen that we have only one car in common. This can happen when the distance between a tracker A from one camera and two trackers, B and C, from another camera are very similar. Thus, sometime A-B will be linked together and at another time it could be A-C. If it is chosen to keep the same global id with only one car in common, it would link B and C with the same id which is a mistake. If it is chosen to keep the same global id with a minimum of two cars in common, it would give tracker A two different id's, which is also a mistake. In conclusion, both choices have their pros and cons. However, if this analysis was done with a cluster of size 4, one common car could create more errors.

Moreover, the hierarchical clustering is only based on one feature, the distance in space, which makes a basic clustering. More robust systems were presented in the state of the art section. It is therefore not the most reliable one. That's why the choice was made to choose at least two trackers in common. This offers additional security to avoid spreading too quickly the errors made by the clustering. With a more complex clustering, the probability of actually having the tracks corresponding to the same car in the same cluster will be higher, it could then be considered taking only one track in common.

All the steps described so far, single camera tracking, clustering and matching, form together an algorithm of MOMCT. The objective of this thesis is not to try to further improve this algorithm. Indeed, as explained in the state of the art chapter, very complex algorithms already exist. Rather, it aims to address a known problem in the field of MOMCT on how to deal with detection errors coming from faulty cameras. With this objective in mind, a feedback step has been added to the algorithm. Before delving into the feedback step, some visualisation of the MOMCT algorithm will be presented.

3.4 Application on AI city challenge dataset

With a provided dataset, it is possible to visualise some results of the implemented MOMCT algorithm.

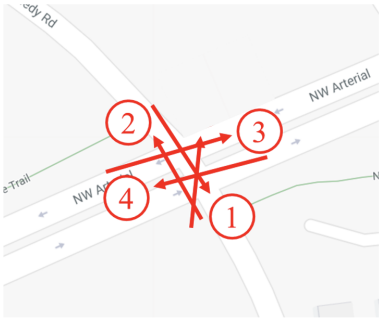


Figure 3.8: A map representing the cameras' localisation on the crossroad *S01* [15].

All results are applied to the AI city dataset [15]. More specifically from *Track 3: City-Scale Multi-Camera Vehicle Tracking*. Most of the analysis of the results will be done on the crossroad accessible in the folder named *S01*. The localisation of the cameras is visible on the map 3.8. In the original dataset, a fifth camera is present. It has been removed due to its distortion. For such a camera, the model explained in subsection 3.2.1, used to project a point on the occupancy map, does not work properly anymore. Another more complex model has to be implemented, which is beyond the scope of this thesis.

Figure 3.9 corresponds to a time frame of the visualisation of crossroad *S01*. Four different points of view, corresponding to the four cameras, are visible. The name of each camera is placed above its corresponding image. Besides, an occupancy map is present in the middle of the figure.

As explained in the clustering step of the methodology, only cars that are fully visible on 2 or more cameras are selected. Thus, the truck that is only visible on camera 4, is not present on the occupancy map. However, the red car is detected by cameras 2 and 4 and should be present. Since the projections on the occupancy map are coming from the same car, they are close to each other. Therefore, they were associated with the same cluster. Furthermore, a new point is computed representing the estimate of the actual position of the car. It is simply the average of all points in the cluster. To dissociate it from the other points, it has a slightly larger radius and the global id of the car is written next to it. The same reasoning can be made with the blue car, giving the second cluster. No other vehicle is visible twice, thus only two clusters are finally represented in the image.

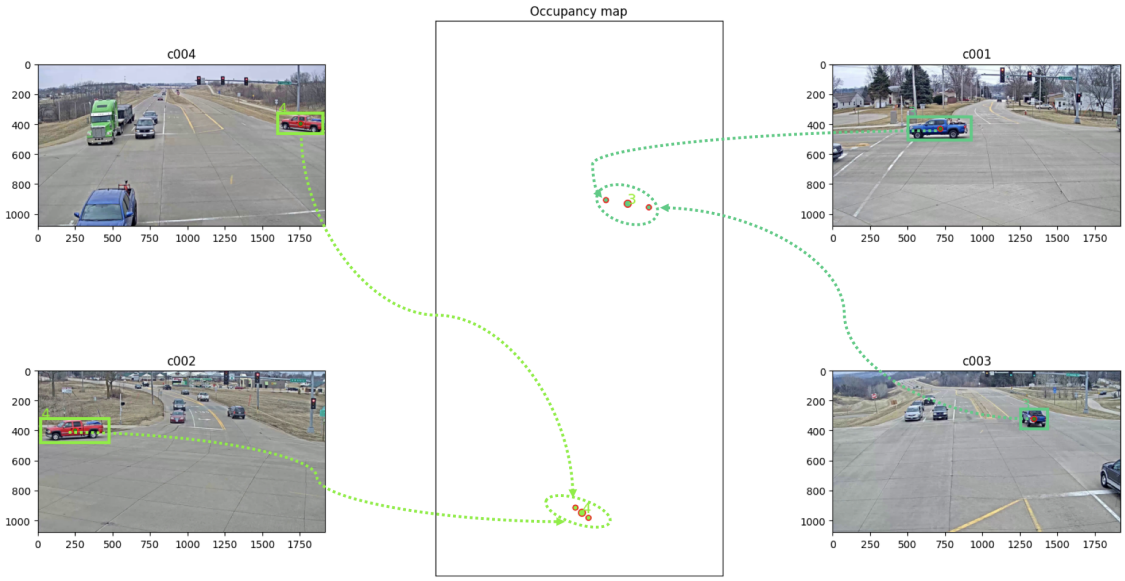


Figure 3.9: Visualisation of the implemented MOMCT algorithm on a crossroad with 4 cameras. Two cars are visible on multiple cameras and are therefore visible on the occupancy map. A last slightly bigger point is present for each car, representing the global estimate position of the car.

The visualisation of this data is necessary for the understanding of the last step, feedback, which is explained in the next section.

3.5 Feedback Reconstruction Algorithm (FRA)

A last step is added to the MOMCT algorithm in order to reconstruct missing detections. The principle is first explained, followed by a visual analysis of its impact.

3.5.1 Principle

The purpose of Feedback Reconstruction Algorithm (or feedback) is to detect an anomaly. It is based on the following reasoning: if two or more tracks are linked together at a certain time frame, they should be linked together at the next one. If this is not the case, a detection is reconstructed from the missing track. Then, three scenarios are possible:

- The reconstructed detection from the missing tracker is in the field of view of the camera. The associated bounding box should be kept.
- The reconstructed detection from the missing tracker is not entirely in the field of view, meaning the car is leaving it. The detection should not be kept. (rules from the ground truth of AI city dataset [15]).
- Another tracker from the same camera is associated with the cluster. This means that the tracker added in the cluster at the previous time frame could be a mistake. In this case, nothing should be done.

Reconstruction process

The reconstruction process will now be described by applying it to an example. Figures 3.10 and 3.11 display the view of each camera at two consecutive times: t and $t+1$ respectively. The images also provide the projections of the detections on the occupancy map. At time $t+1$, in figure 3.11, camera 3 (c003) misses the detection of the car. Therefore the FRA is activated. The process can be explained in 4 main steps:

- Find the old projection of the missing car at time frame t.
- Compute a distance relative to another projection of the same car at time t (blue line on figures 3.10 and 3.11).
- Compute the new projection point of the faulty camera at time t+1 (red point on 3.11) based on the relative distance (blue line) found.

- Project the computed point backwards on the camera image, giving you the middle point of a new detection (red detection on camera 3 in figure 3.11).

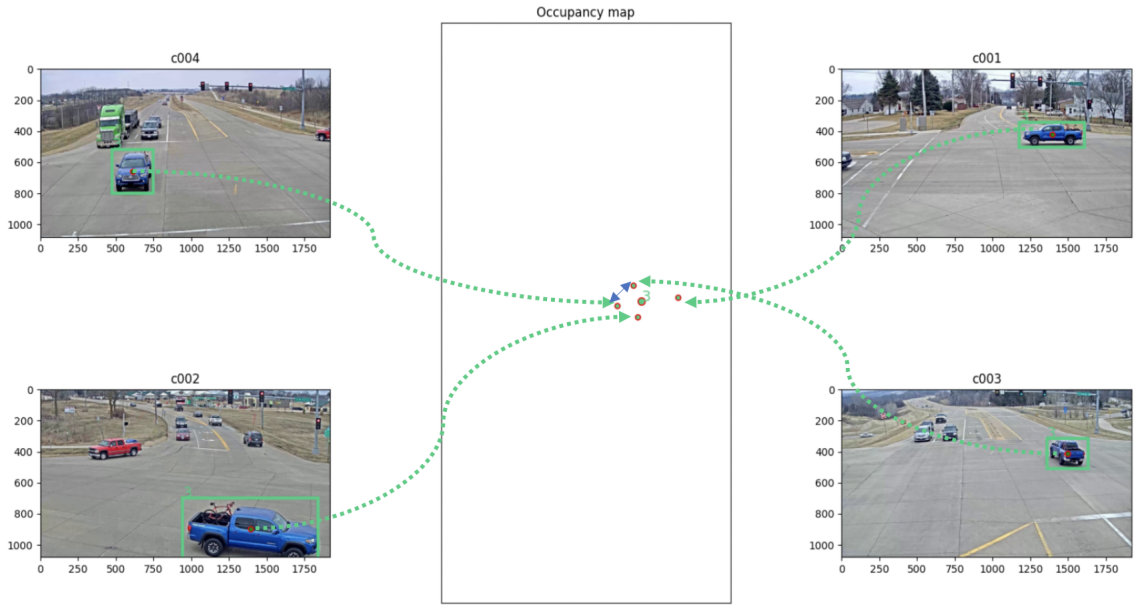


Figure 3.10: Visualisation of the implemented MOMCT algorithm at a crossroad with 4 cameras at time frame t. One car is visible on all cameras and is therefore projected 4 times on the occupancy map. The blue line represents the distance between the projection of camera 4 and the projection of camera 3 of the same blue car.

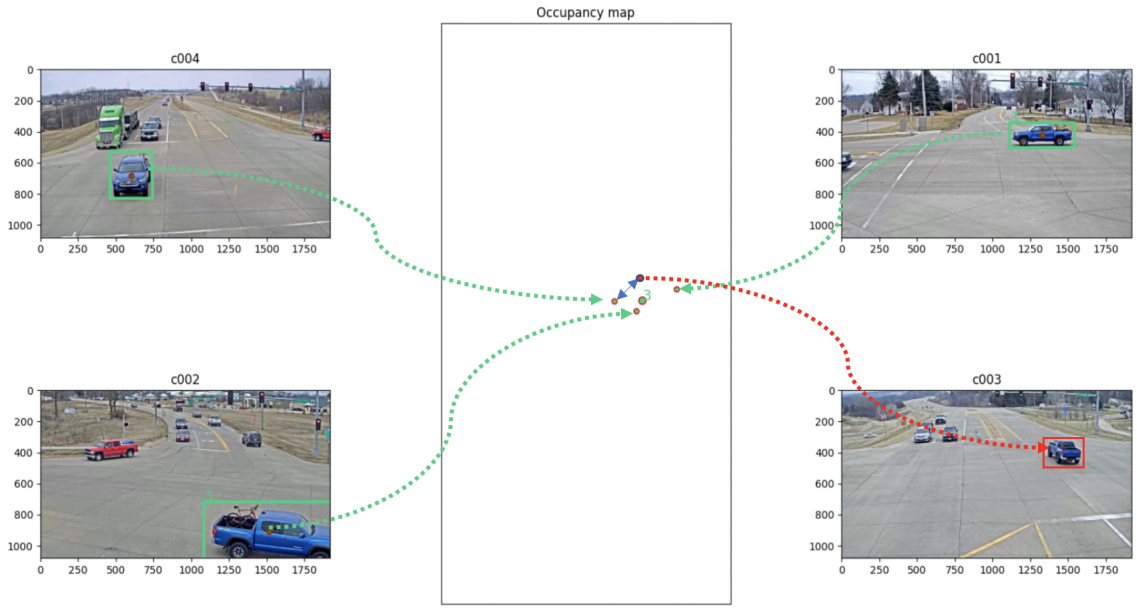


Figure 3.11: Visualisation of the implemented MOMCT algorithm on a crossroad with 4 cameras at time frame t+1. One car is visible on 3 cameras. Camera 3 is faulty and its detection is recovered by the FRA.

The method will now be explained in a more mathematical way with the help of image 3.12. It shows the occupancy map of figures 3.10 and 3.11. The objective is to estimate the new position, at time t+1, of the missing detection coming from the faulty camera 3. At time t, cameras 3 and 4 project their detection at points (x,y) and (x+a,y+b) respectively. The relative distance between the two projections is (a,b). This represents the projection error between the two cameras. Thereafter, at time frame t+1, the vehicle's position has slightly moved. The detection of camera 4 is projected at point $(x + \epsilon, y + \gamma)$. The projection of the missing detection is estimated to be at $(x + a + \epsilon, y + b + \gamma)$, using the relative distance (a,b) found at the last step.

Once the missing detection point has been created, a final step consists in projecting it back into the faulty camera pixels coordinates. This is done using the inverse of the homography matrix H in equation 3.8 described in 3.2.1. The result is the middle point of the bounding box on the faulty camera. For the dimension of the bounding box, the same is used as the one detected at the previous time frame.

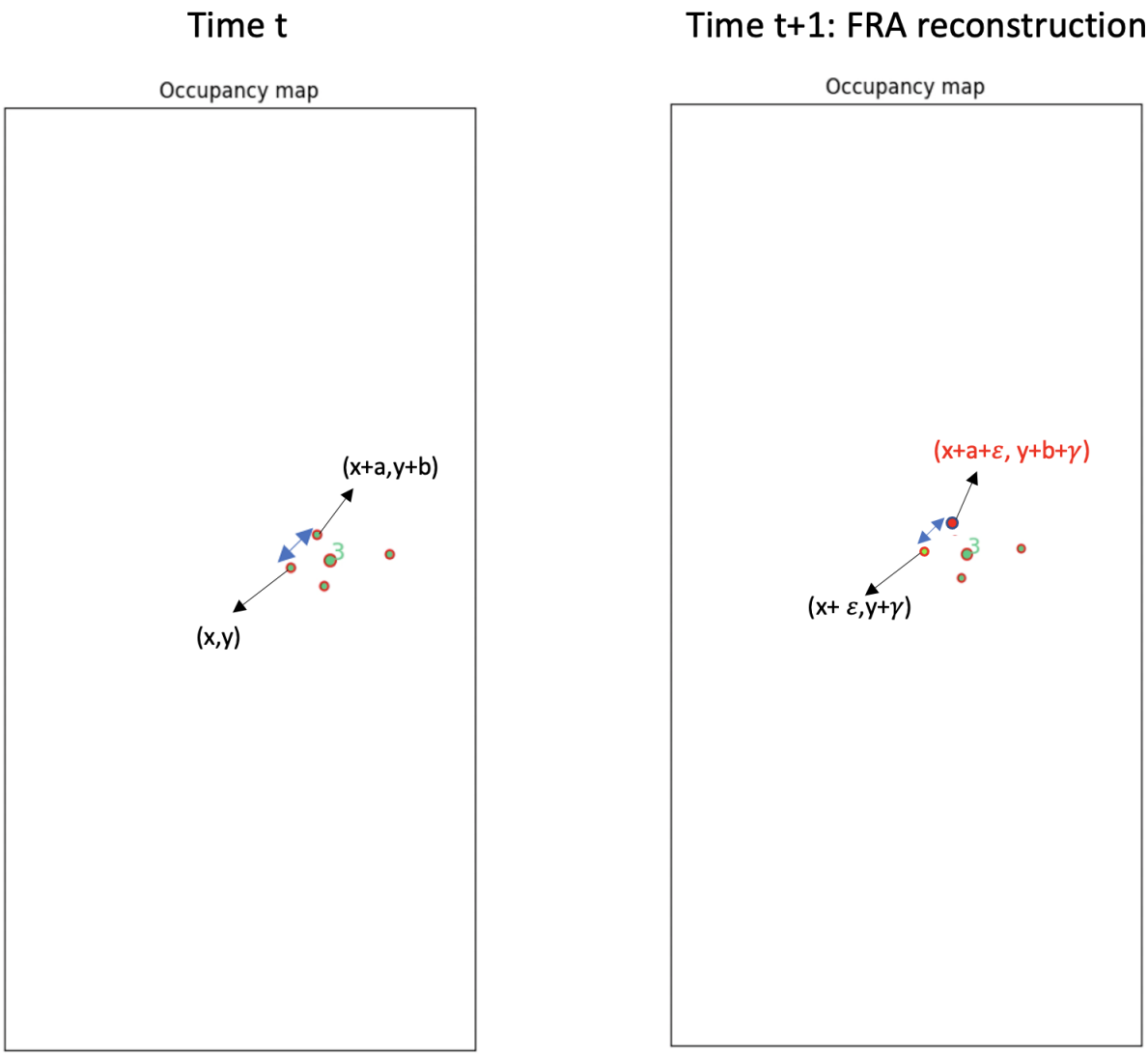


Figure 3.12: Reconstruction process of the FRA. A new point $(x + a + \epsilon, y + b + \gamma)$ is estimated from information at the previous time frame and another detection at the same time frame.

During each projection a small error is made, this is why distance (a,b) is perceptible. It is the bias of the homography matrix. Only one other projection is used in order to estimate this bias. Since the time difference between the two frames is very small (the order of 0.6s), using only one other projection is already quite accurate. This projection is chosen randomly.

3.5.2 Visual analysis

A first simple approach to validate the functioning of the feedback is by analysing its impact visually. Figure 3.13 shows three different scenarios.

- Scenario 1: There are no missing detections. Both cameras track the car giving it the same id all the time.
- Scenario 2: The feedback is inactive. Camera B misses one detection. When the car is detected again, a new track is created with a new id, suggesting to be a different car than the one previously detected.
- Scenario 3: The feedback is active. Camera B misses one detection. The feedback reconstructs the missing detection. The track of the car is not lost and no new car id is created.

This is a first demonstration of the usefulness of the feedback.

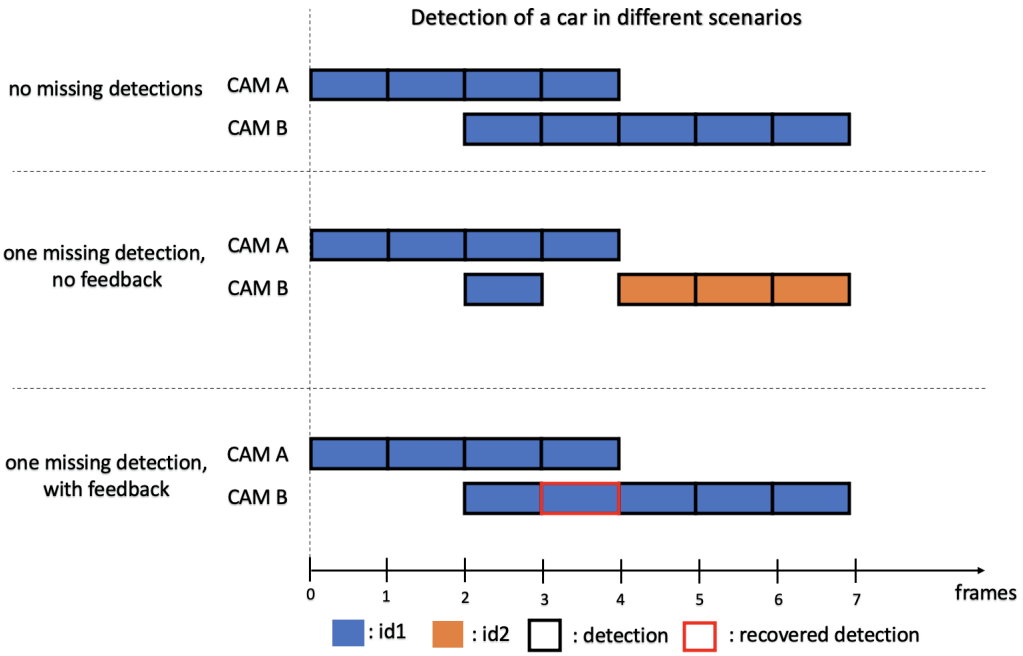
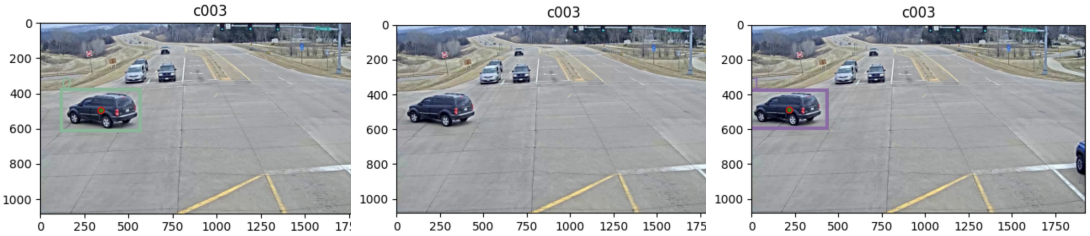


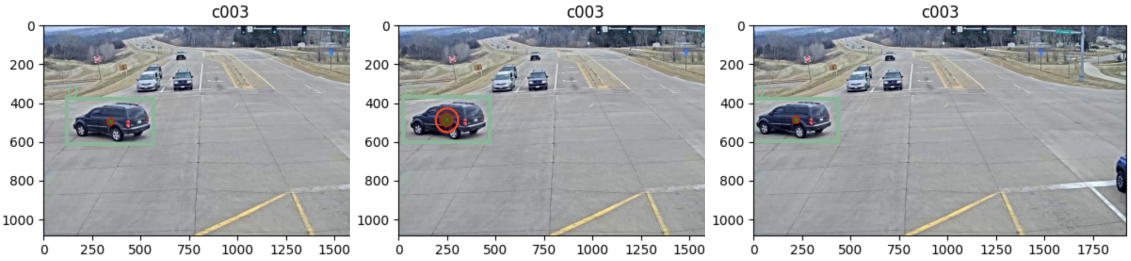
Figure 3.13: Detection of a car by 2 cameras, A and B, during 7 time frames. Three different scenarios are presented, showing the usefulness of the feedback.

The same effect can be observed on data provided by AI city challenge [15]. Figures 3.17 and 3.21 correspond to camera B in scenarios 2 and 3 respectively. A change in id corresponds to a change of color of the bounding box. A recovered detection from the feedback is represented by a bigger center point in the corresponding bounding box.



frame 2, (a) frame 3, (b) frame 4, (c)

Figure 3.17: 3 consecutive frames of camera 3 tracking cars without the feedback activated. Camera 3, being partially faulty, missed one detection on frame 3. The tracking is therefore interrupted which leads to a change in the car's id in frame 4.



frame 2, (d) frame 3, (e) frame 4, (f)

Figure 3.21: 3 consecutive frames of camera 3 tracking cars with the feedback activated. Camera 3, being partially faulty, misses one detection on frame 3. The feedback recovers it. The recovered bounding box is represented by a bigger center point. As a result, the track is not lost and the global id remain the same.

Although the feedback step can recover some of the detections, it cannot recover a non-functioning camera. Indeed, the feedback mechanism is based on previous detections by the faulty camera. Consequently, If no detections are made, no detections can be recovered.

3.5.3 Conclusion

Summing up, the feedback is an additional step aiming to recover missing detections. It uses information from the previous time frame in order to reconstruct information for the subsequent time frame. The goal is to increase the efficiency of the tracking algorithm when detections are missing.

The next stage is to assess the effectiveness of the feedback. This is achieved by using the HOTA metric, which is presented in the next section.

3.6 HOTA Metric

Several metrics are available for evaluating multi-object tracking performance. Four of them are used frequently: HOTA, MOTA, IDF1 and Track mAP. Since both detection and association are very important, the metric should weight each of them equally. HOTA has the best balance and is therefore chosen.

According to article [13]: *The MOTA score is heavily biased towards measuring detection at the expense of ignoring association. In contrast, the IDF1 score is heavily biased towards measuring association, at the expense of ignoring detection. HOTA finds a perfect balance between these two extremes by equally weighting both detection and association. Finally, the Track mAP metric is the least informative of all because it requires predicted tracks to have more than 50% overlap.*

The HOTA score is a combination of three different sub-metrics: $LocA, DetA_\alpha$ and $AssA_\alpha$. Each sub-metric will evaluate a different aspect of the tracking (detection, localization and association).

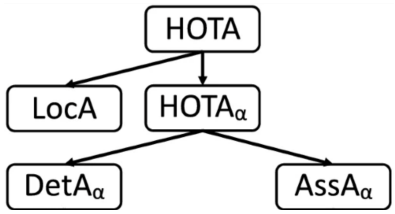


Figure 3.22: HOTA score: a combination of 3 IoU scores. Image provided by [13].

Each of them is an Intersection-over-Union (IoU) score. The principle of IoU is explained in figure 3.22. To sum it up briefly, IoU is the area of overlap divided by the area of the union of two bounding boxes. One bounding box comes from the predicted detection and the other one comes from the ground truth. The assessment covers the following aspects:

- Localization: Measures how well a predicted bounding box matches a ground truth one in space. The focus is therefore on the location of each bounding box.

- Detection: Compares the set of predicted bounding boxes with the ground truth ones. The focus is more on the number of detections rather than the individual position.

- Association: Compares the association over time of id's from the predicted set with the ground truth ones.

The benefit of having three sub-scores is to be able to understand which part can be improved, and therefore know which variable should be tweaked. Or the other way around, it enables to identify more precisely the effect of a variable when it is tweaked. However, in this thesis, we only use the metric for evaluation purposes. Hence, only the global HOTA score is used.

This score is a multi-object tracking score, which corresponds to a score for a single camera. This is a perfect metric for an algorithm such as SORT but has to be adapted for a MOMCT system. Nevertheless, since the only detections kept in the predictions are the ones coming from a cluster of minimum size two, the clustering is indirectly evaluated within each camera score.

However, interchanging the global id of one car with that of another will has no effect on the score. This means that the consistency of id's across different cameras has no impact on the score. To tackle this issue, it is necessary to combine all selected detections from all cameras into one single file. Each time a camera is added to the file, its time-frames are modified to make it seem as if everything was filmed by the same camera. This is a little trick for having a general HOTA score that checks the consistency across different cameras. This idea was suggested in article [6]. Eventually, this method computes a global HOTA score.

Finally, the code used to compute the HOTA score is the same as the official evaluation code from different benchmarks such as the MOTCHALLENGE, RobMOTS, Open World Tracking and many more. It is called TrackEval and is available on the site [12].

Chapter 4

Results and discussion

In this chapter, the analysis will go further than a visual validation of the feedback. Thanks to the HOTA metric described in the previous chapter, the efficiency of the feedback method can be assessed numerically.

Since the focus of this thesis is on the effectiveness of the feedback step, its impact has to be evaluated. As a reminder, the purpose of the feedback is to reconstruct some missing detections. The idea is therefore to remove a certain percentage of detections from a camera and to compare the score with and without feedback. The expected result would be a higher HOTA score with feedback, thanks to the reconstruction of a certain amount of the missing detections.

The analysis consists of several parts. Firstly an in-depth analysis will be carried out on one crossroad of the IA city dataset [15]. Then, in order to ensure that the results found are not specific to a single intersection, a comparison will be made with the results on a second intersection. Subsequently, a temporal performance analysis of the program is also performed to ensure a real-time performance. Finally, the key points are listed in the conclusion section.

The code used is available at `https://github.com/pierrecaprasse/fault_tolerant_tracking`.

4.1 Analysis on a crossroad

To measure the FRA's effectiveness on a single crossroad, several tests are made. The first one tests its impact when all cameras are working perfectly well. Next, its effectiveness is tested when a faulty camera is present. To do so, a certain amount of detections made by the detector of the faulty camera are deleted. This test is repeated several times for different deletion percentages, growing linearly from 0 up to 100% (i.e. suppressing the totality of detections). The test is performed for each camera individually and results are compared. A last test is performed were some detections are deleted on all cameras simultaneously.

4.1.1 Effect of FRA without faulty camera

First, it is imperative to check whether or not the feedback has a negative impact when all cameras are working properly. Since this is supposed to be the most frequent situation, the feedback would lose its usefulness if it has a negative impact in this situation. Indeed, since no detections are missing, the feedback is not supposed to be activated. Therefore, the results are supposed to be identical. They are visible in table 4.1.

	HOTA without FRA	HOTA with FRA
Global Score	47.35	47.679
CAM 1	49.395	49.564
CAM 2	58.13	58.548
CAM 3	48.695	49.288
CAM 4	55.147	55.132

Table 4.1: HOTA scores of the different cameras when no partial failure is added. Two versions of the MOMCT algorithm are compared: with and without the FRA step.

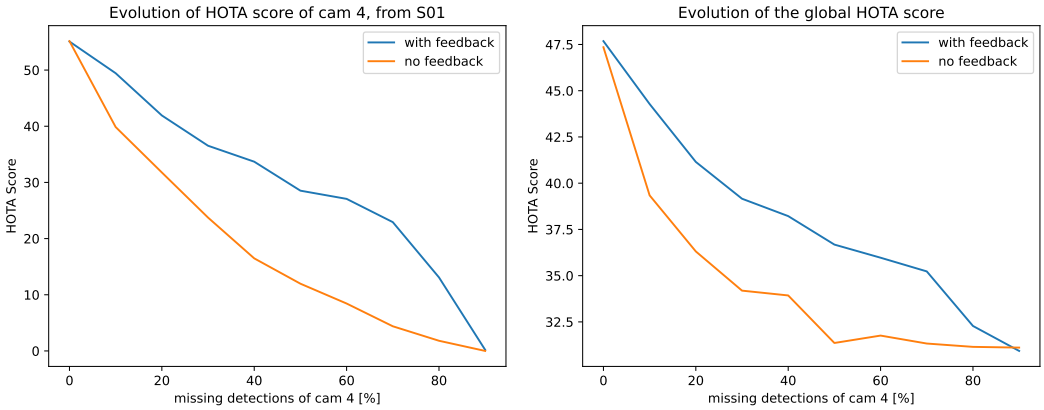
As expected, **the results are very similar** with and without the FRA. For camera 4, the score has slightly decreased. This can occur due to errors when a car is very close to the edge. If the original detection is just not entirely on the image, it should not be selected (rules of the ground truth of the AI city dataset). However, this will activate the feedback since it is a missing tracker in a cluster. If the reconstructed detection by the FRA it slightly off and therefore entirely on the image, it will be selected. This decreases the score. For the other cameras, the score has slightly increased. A possible explanation is the reconstruction of a detection that was not matched when it had been matched in previous frames.

The scores from different cameras can also be compared. They vary between 48 and 58. Cameras 1 and 3 scored slightly lower than cameras 2 and 4. A possible explanation would be that cameras 2 and 4 have a rather similar field of view and a rather similar bias in the homographic projection. This would lead in having a higher number of common clusters, which would increase their individual score. As a reminder, each selected detection is present on minimum two cameras. The location of the cameras relative to one another reinforces this hypothesis. Indeed, on figure 3.8, cameras 2 and 4 are close to each other and share a large common field of view.

Finally, **the global score is lower than the worst camera**. As mentioned in 3.6, in addition to computing the individual score of each camera, the global score also checks the consistency of id's across them. Hence, two cars that have been interchanged will lower the score. The global score is therefore lower and in this case is equal to 47.679. It is a little lower compared to the state of the art. Yet, it is still a reasonable score. Again, the objective of this thesis is not to optimise the MOMCT but rather to figure out the impact of the FRA. In this section, we have just shown that it has no negative impact when all cameras are working normally.

4.1.2 Main impact of the feedback

The next goal is to measure the impact of the reconstructed detections from the FRA on the HOTA score. To do so, in a first instance, camera 4 is artificially supposed to miss detections. Therefore, detections from camera 4 are deleted sequentially, going from 0 up to 100%. Camera 4 was chosen first because the results have the typical expected outcome. More specific cases will be described later. The results are visible in figure 4.1. On image (a), two curves are visible, both from the HOTA score of camera 4. One represents the evolution of the HOTA score of the MOMCT algorithm using the FRA step and the other one without.



(a) (b)

Figure 4.1: Evolution of the HOTA score of camera 4 on (a) and of the global HOTA score on (b) depending on the percentage of missing detections by camera 4. Two methods are compared, with and without the FRA step.

Unsurprisingly, the HOTA score without feedback is decreasing when the number of missing detections is increasing. Indeed, fewer detections means fewer correct detections, resulting in a lower HOTA score. Positively, the curve representing the algorithm including the **feedback performs globally better than the one without**. This implies that the feedback did recover a certain amount of deleted detections. The optimal curve would have been a horizontal line, meaning that all missing detections were recovered. However, given the implementation of the algorithm, not all detections can be recovered. For instance, if the first detection is suppressed, no previous detection is available to activate the feedback. It will therefore never be recovered. This brings us to the last comment, if 100 % of the detections are deleted, none will be recovered. The HOTA score is then zero for both algorithms, which can be seen on the graph.

The decrease of the HOTA score of camera 4 compared to the missing detections is reflected in the global HOTA score visible on image (b) of 4.1. The curve with feedback is also higher than the one without. Note that the scale of the HOTA score axis is different compared to image (a). When camera 4 is completely out of order, the global score decreases from 47.679 to approximately 28. The remaining score comes from the other cameras that are still working. This is the reason why the HOTA score of camera 4 decreases to 0 while the global HOTA score does not.

On image (a) of 4.1, at 60% missing detection rate on camera 4, the FRA increases the score from 8 to 27. Knowing that the initial score was equal to 55, the system recover 40% of the HOTA score. The same reasoning can be made on image (b), recovering 37.5% of the global HOTA score of the intersection.

Besides, the first points of the curves correspond to the scores in table 4.1.

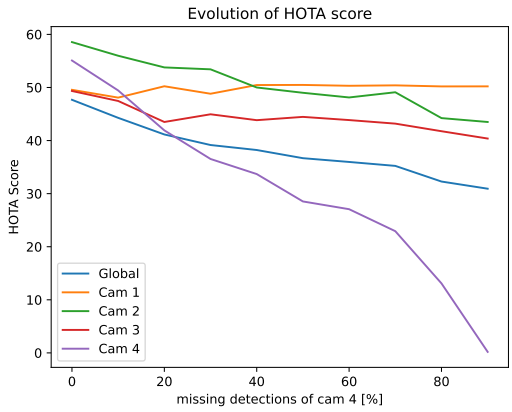


Figure 4.2: Evolution of different HOTA scores of the algorithm with feedback depending on the percentage of missing detections by camera 4.

A final analysis can be made to assess the impact of one camera on the others. In figure 4.2, all scores, the global one and the ones from each cameras, are compared. Here only the algorithm using the feedback is represented. Thus the purple curve, representing the score of camera 4, is the same curve as the blue one on image (a) of figure 4.1. In the same way, the blue curve is the same as the blue curve on image (b).

The purpose of this figure is to visualise the impact of one camera on the others. For instance, we can see that the score for camera 1 is only slightly impacted by the change of detections of camera 4, whereas camera 2 is much more impacted.

4.1.3 Two additional observations

The same methodology is now applied on camera 1 instead of 4. The results of this scenario are visible in figure 4.3. Without any surprise, the shapes of the curves are very similar to the ones in the previous scenario. However, two additional observations can be made.

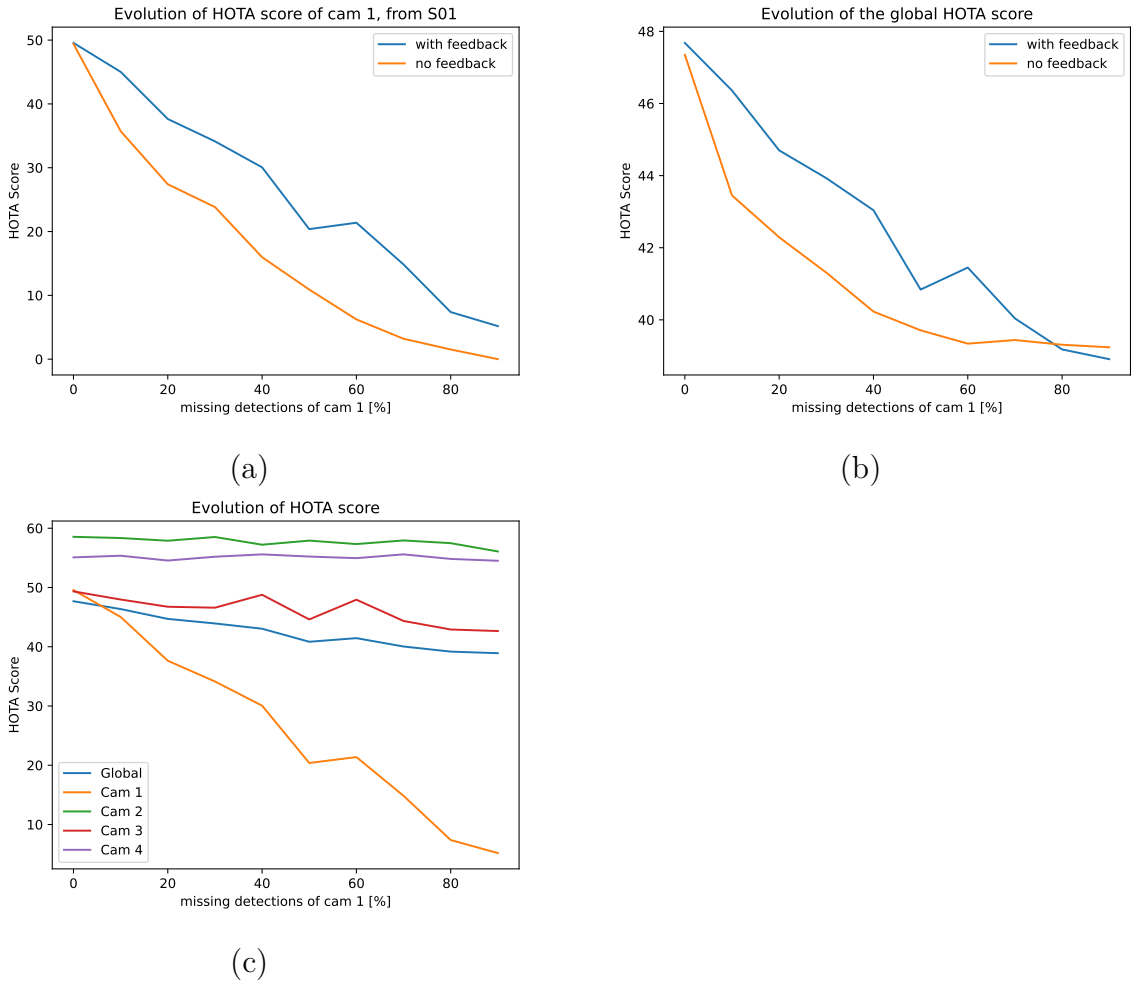


Figure 4.3: Evolution of different HOTA scores depending on the percentage of missing detections by camera 1. On (a), the evolution of the score of camera 1 with and without feedback, on (b) the evolution of the global score with and without feedback and on (c) the evolution of HOTA scores for all cameras using the feedback.

Counter-intuitive increase

First, on image (a), the curve with feedback rises when the percentage of deleted detections increases from 50 to 60%. This is counter-intuitive. A plausible explanation is as follows: at 50%, a car is first matched in the right cluster and then matched several times in a cluster corresponding to another car. If at 60%, the detections that are matched in the wrong cluster are deleted, the feedback will also be activated for those missing detections. Since the first detection was in the right cluster, the feedback will then reproduce detections and add them to the right cluster. The HOTA score increases as a result. This increase is also reflected in the overall score on image (b).

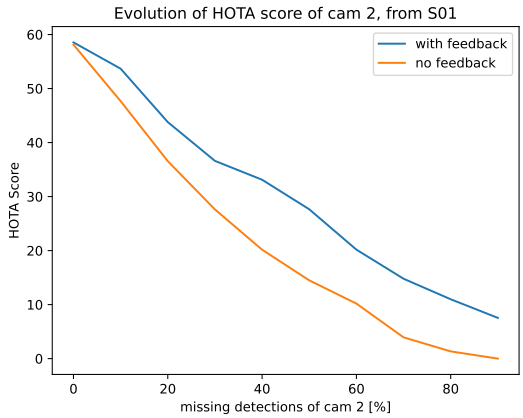
Relationship between the cameras

The second comment concerns image (c). There are almost no changes in the curves of cameras 2 and 4. On the other hand, camera 3 follows a pattern rather similar to the one of camera 1, only less pronounced. This suggests a connection between cameras 1 and 3.

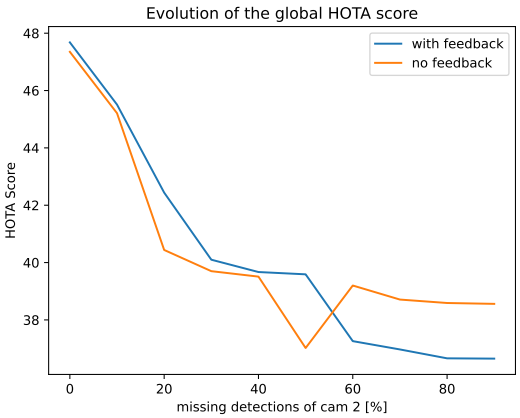
We remind that each selected cluster contains at least two detections. In other words, each car is selected if and only if it is visible on two cameras. The analysis in the previous subsection has shown a strong relationship between cameras 1 and 4. Here, we find a strong relationship between cameras 2 and 3. Related cameras have thus several tracks in common, which implies a great overlapping field of view.

4.1.4 Faults in the leading camera

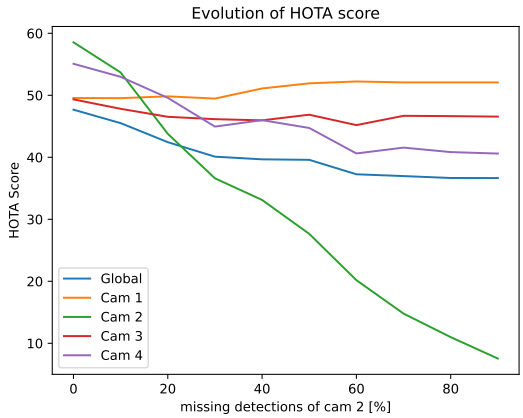
The results for camera 2 are visible in figure 4.4. The results for its own score on image (a) are similar to the ones achieved by the other cameras and presented in the previous sections.



(a)



(b)



(c)

Figure 4.4: Evolution of different HOTA scores depending on the percentage of missing detections made by camera 2. On (a), the evolution of the score of camera 2 with and without feedback, on (b) the evolution of the global score with and without feedback and on (c) the evolution of all HOTA scores using the feedback.

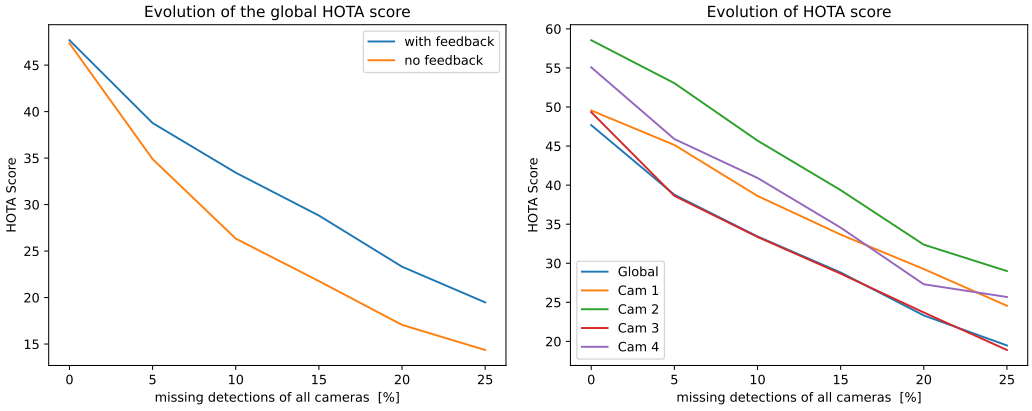
However, the global score is worth analysing in more details. Camera 2 is the best functioning camera because it has the best HOTA score without missing detection. It has therefore a greater impact on the overall score when it is not functioning properly. This is especially the case, since camera 2 has a great impact on camera 4, which is the second-best camera. Moreover, analysing graph (c) shows that the abnormal dips on the global score curve are rather similar to the abnormal dips in the curve of camera 4 and can therefore be attributed to camera 4. To conclude, the shape of the overall score curve is mainly influenced by the curves of cameras 2 and 4, since the curve of camera 4 is the most impacted by the relationship between both cameras.

A similar analysis has been performed for camera 3. However, no new teaching has been brought to light. Hence, the results are not presented in this section but can the figures can be found in appendix C.

4.1.5 Simultaneous errors

In this section, a final scenario was carried out, where a certain amount of detections are deleted from all cameras simultaneously. The results can be viewed on 4.5. In this scenario, the percentage of deletion goes from zero to 25 % and not 100 %. The reason is that detections are deleted on each camera, while there were deleted on only one single camera at a time in the previous sections.

The FRA works as expected for each camera. It performs better when the feedback is used than when it is not. This result is valid for all camera scores and irrespective of the percentage of missing detections we are looking at. Thus, the global score has the same shape as the individual HOTA scores and is visible on image (a) of figure 4.5.



(a) (b)

Figure 4.5: Evolution of different HOTA scores depending on the percentage of missing detections by all cameras. On (a), evolution of the global score with and without feedback, on (b) the evolution of all HOTA scores using the feedback.

Note that when removing the full 25 % of detections on each camera the score drops to approximately 20% when the feedback is used. When the same number of detections is removed on a single camera (100 % of the detections), the global score is still around 35 %-40% . This means that **the situation is worse if all cameras are only partially functioning rather than one camera not functioning at all.** .

4.2 Generalization analysis

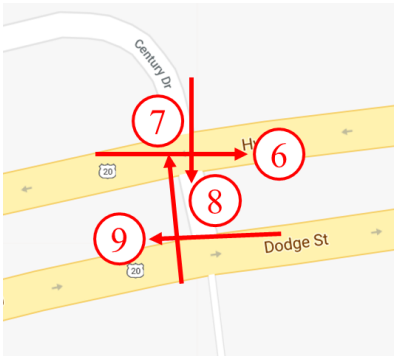


Figure 4.6: A map representing the camera localisation on the crossroad *S02*. [15].

All results so far have been applied to the same crossroad. In order to generalise the results, a similar analysis is performed on a **second intersection** of the AI city dataset [15]. It can be found in the folder *S02* and its camera locations can be viewed on figure 4.6.

The score of the algorithm when no failure is perceived can be found in table 4.2. The **scores hover around the same values** as in table 4.1 even if they are globally slightly lower. The algorithm with feedback is again as robust or better than the one without. The conclusion is therefore the same as for the first junction: the feedback has no negative impact when all cameras are working perfectly well.

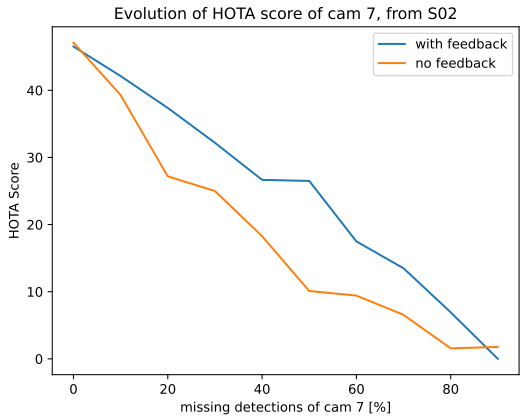
	HOTA without F	HOTA with F
Global Score	39.581	40.602
CAM 6	56.923	59.2
CAM 7	46.6	47.104
CAM 8	27.225	28.438
CAM 9	35.611	36.248

Table 4.2: HOTA scores of the different cameras when no partial failure is added. Two versions of the algorithm are compared: with and without the feedback step.

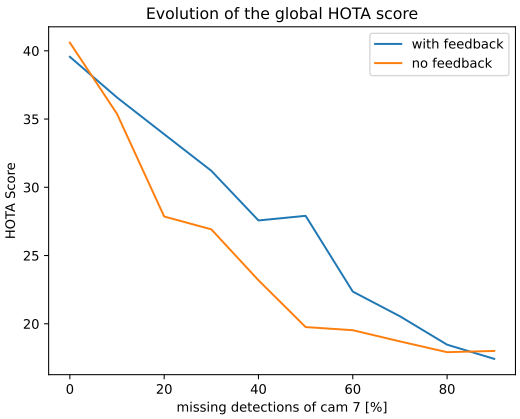
The graphs corresponding to the simulation of the increase in the malfunctioning of camera 7, chosen randomly, are visible in figure C.2. The shapes of the curves are as expected. The scores decrease when the number of missing detections increases. The curves representing the scores using the feedback are higher than the ones without feedback, thanks to the reconstructed detections.

Finally, in image (c) it can be seen that the missing detections of camera 7 have an influence on all other cameras except the 8. This can again be explained by the lack of common field of view between both cameras. Camera 7 is looking in the direction of camera 8. It can therefore see everything that is happening behind

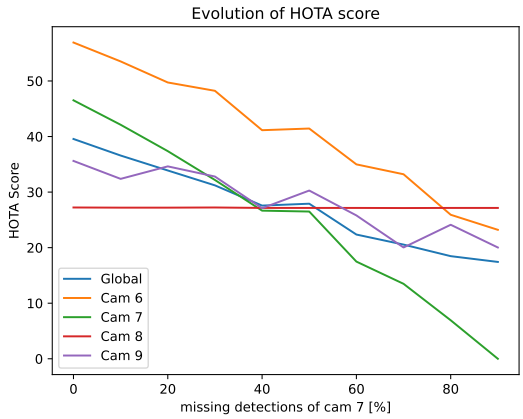
this camera and what is happening just in front of it, which is hardly visible for camera 8. Thus, there is almost no common field of view, which implies almost no common detections. This in turn involves that the results of one camera have no impact on the results of the other.



(a)



(b)



(c)

Figure 4.7: Evolution of different HOTA scores depending on the percentage of missing detections by camera 1. On (a), the evolution of the score of camera 1 with and without feedback, on (b) the evolution of the global score with and without feedback and on (c) the evolution of all cameras' scores using the feedback.

In summary, the results are very similar for the crossroad *S02* analysed in this section and for the crossroad *S01* analysed in the previous section. In both cases, the feedback reconstructs part of the missing detections when a camera is faulty. This improvement is reflected by a higher HOTA score.

4.3 Program temporal performance analysis

Another objective of the tracking algorithm is that it can be used in real-time. It is therefore important to analyse its execution time. The arrival time between each image and the execution time for one image have to be compared. If the execution time for one image is less than the time between the arrival of two frames, the algorithm can be run in real-time. Otherwise not. In principle, the code should be able to run in real-time as none of the algorithms used have very high complexity. For instance, no supervised learning are used.

The frame rates of all videos of AI city are 10 Frames Per Second (FPS). The crossroads in the file *S01* contains 1,955 frames. Knowing the FPS, this corresponds to 195.5 seconds of video, which is equivalent to 3.2667 minutes. Since the recording takes place in the middle of the day with traffic, it is already a reasonable time to find an estimate of execution time.

Several tests were performed. The code was run with feedback, without feedback and with a 30% suppression for camera 3. The execution time was also computed on the second crossroads. The results are available in table 4.3 in seconds and in table 4.4 in FPS. Tree different times are presented:

- Execution time for the single camera tracking in the column *SCT*.
- Execution time for the inter-camera tracking which includes clustering, matching and feedback under the column *ICT*.
- Total execution time, which is just the sum of SCT and ICT, is in the last column.

Note that execution time varies from execution to execution and therefore cannot be perfectly reproduced. For this reason, each test was run 5 times and the average execution time is presented.

Crossroad	frames	feedback	Error Cam 3	SCT (s)	ICT (s)	TOTAL (s)
S01	1955	ON	/	4.930	41.798	46,728
		OFF	/	4.813	42.481	47,294
		ON	0.3	4.557	29.457	34,014
S02	1924	ON	/	4.744	34.670	39,414

Table 4.3: Temporal performance analysis in seconds. Time needed for single camera tracking (SCT), inter-camera tracking (ICT) and total time of different executions can be found.

Crossroad	frames	feedback	Error Cam 3	SCT (FPS)	ICT (FPS)	TOTAL (FPS)
S01	1955	ON	/	396	47	42
		OFF	/	406	46	41
		ON	0.3	429	66	57
S02	1924	ON	/	405	55	49

Table 4.4: Temporal performance analysis in frames per seconds (FPS). The FPS needed for single camera tracking (SCT), inter-camera tracking (ICT) and total FPS of different executions can be found.

The **main findings** that can be drawn from the tables are:

- Execution time for the SCT part (around 5 seconds) is much shorter than ICT time (around 40 seconds).
- Execution time with and without feedback are very similar.
- Deletion of detections makes the ICT part less complex and therefore faster. Conversely, this implies that if traffic density increases, the number of detections will increase,thereby increasing execution time.
- Execution times are quite similar at both crossroads.
- Finally, speed performance of the algorithm using the feedback with no faulty camera is 42 FPS, which is significantly faster than the 10 FPS generated by the cameras. Hence, the algorithm can be used for real-time tracking. With this margin, an increase in traffic density should not effect this conclusion.

4.4 Conclusion

Thanks to the use of the HOTA metrics, this section confirms the usefulness of adding a feedback step (FRA) to improve tracking performance at a crossroads by reconstructing missing detections from faulty cameras.It is first established that using the feedback does not impact negatively the use of the system when cameras are working properly. More importantly, it shows that the feedback step enables to recover part of the missing detections when a camera is faulty, thus resulting in a positive impact on the overall score. Nevertheless, not all detections are recovered and when a camera is totally out of service, the feedback becomes useless. Besides, when a camera misses some detections, it has a negative impact on other cameras sharing large common fields of view with the faulty one. These observations are observed at two different crossroads in order to strengthen their validity. Finally, a temporal performance analysis is performed. It shows that the code and feedback could run in real-time provided the image speed does not exceed 40 FPS.

Chapter 5

Perspective

A MOMCT algorithm was presented and implemented in this thesis. Then, a new solution was proposed to correct errors coming from faulty cameras, called interchangeably Feedback Reconstruction Algorithm (FRA) or feedback. Based on the results obtained by simulation, the FRA showed to be effective in reconstructing missing detections. Several steps could be undertaken to further this analysis. Three main directions are suggested.These suggestions are not intertwined, nor are they mutually exclusive. So only some of them could be implemented and each of them could be undertaken separately. The first section of this chapter proposes an improvement of the MOMCT algorithm based on related works. The second section presents two additional tests that could performed on the implemented algorithm. Lastly, new ways of improving and using feedback are presented.

5.1 MOMCT improvement

Several ways to improve clustering can be found in the state of the art. The main points that could be added are:

- Replace SORT by deepSORT in the single camera tracking (SCT) step [9] [18] [3] .
- Add appearance features for vehicle recognition with a convolutional neural network (CNN) [9] [18].
- Use a linear prediction model in the clustering step. It could simply be done with the kalman filters already available [9].
- Implement a Weighted aggregation methodology for combining different features (appearance features, homographic distances, etc.) [9] [18].

- Optimise matching using a template matching strategy. [18]
- Optimise the implementation of hierarchical clustering with a priority queue [9].

All these methods have already been implemented in related works. One could simply take one of the best MOMCT codes currently available and add the feedback step.

5.2 Additional tests

5.2.1 Scalability test

A scalability test can be performed on a crossroads to measure its ability to scale up. This test was not performed for various reasons related to the database used [15]. The crossroads with the largest number of cameras available includes 5 cameras. Therefore, it is impossible to test scalability with a larger number of cameras on this dataset. Besides, since each car in the ground truth has to be visible by at least two cameras, the ground truth depends on the number of cameras installed at a junction. The ground truth is therefore modified when the number of cameras changes, which makes it difficult to compare the information added by a new camera. However, such a ground truth could better assess the inter-camera component.

To conclude, a scalability test can be performed if a dataset is available with a higher number of cameras at the crossroads and a more suitable ground truth. A more suitable ground truth for a scalability test would be to have all cars present within a certain perimeter.

5.2.2 Miss-placed detections test

Another kind of error that can occur is a misplaced detection. This could be simulated by adding white noise to some detections of a camera. If the inter-camera matching works correctly, the tracker should not be associated with any other tracker since the tracker has a different feature due to the white noise. This should also activate the feedback. This should be tested with the improvements suggested in the previous section.

5.3 Future use of feedback

5.3.1 Recovered detections to improve SORT

The FRA produces detections that are inserted in the output and used at the next time frame in the matching step. In addition, it could also be used to improve the single camera tracking algorithm SORT in two different ways. First, it could be added to the corresponding Kalman filter that was unmatched before the feedback. Second, it could be used to efficiently remove trackers. In SORT, a tracker is removed if it is unmatched T_{lost} times in a row. T_{lost} is a fixed parameter. However, it would be better to remove it only when the car leaves the camera's field of view. This case is often detected by the feedback. Therefore, the deletion of a tracker could in some cases be automated and no longer be linked to a fixed parameter chosen by a human.

5.3.2 New activation function of feedback

Instead of activating the feedback based on matching done at the previous time frame, it could be based on all matching already performed.

Assume that at time frame t, Camera 1 projects a car's tracker to point (x,y). Another camera projects the same car at point (x+a,y+b). Assume the clustering matched the two tracks together. A mapping between the two points could be made by saving this distance (a,b) in a matrix.

If later, another car passes by the same place, it should be detected by camera 1 and projected again at point (x,y). Given the scenario at time frame t, the car is also supposed to be detected by the other camera at the position (x+a,y+b). If this is not the case, the feedback could be activated. This is an example of how past information could be used to activate the feedback instead of only looking at the previous time frame. Once this matrix is populated, we are supposed to be able to reconstruct any detection regardless of the camera's state.

To sum up, this process would create a matrix that, depending on the position of a detection, would indicate whether another detection was supposed to take place on another camera and where it should take place. It would be based on the matching that took place in the past.

5.3.3 Implementation on real crossroads

The results are based on executions made on a personal computer using the AI city dataset [15]. The data are real and not virtual images which already encompasses some of the problems encountered in a real situation. However, several other issues can be encountered when setting up the system at a real crossroads using hardware devices. Other students from UCLouvain wrote a thesis on a scalable architecture for multi-tracking problems using several cameras on Raspberry Pi [1]. In addition, article [11] proposes a communication protocol to reduce the transmission bandwidth. These two papers could be used to implement the system at a real junction.

Chapter 6

Conclusion

This thesis presents a novel system to reconstruct missing detections within the context of large-scale traffic flow analysis. It contributes to partially solve issues caused by heavy occlusions or malfunctioning cameras. For this purpose, the first step is to implement a simple MOMCT. It consists of three main steps: single camera tracking, clustering and matching. Then, an additional step is added to recover missing detections, called Feedback Reconstruction Algorithm (FRA) or feedback. It is the main focus of the thesis. The FRA recovers missing detections of an object thanks to redundant information generated by other cameras detecting the same object.

Several simulations are run on crossroads from a dataset provided by the AI city challenge [15] and assessed using the HOTA metrics. Results of these simulations confirm the usefulness of the FRA. It is first established that the HOTA scores are not influenced by the use of FRA when all cameras are working properly. In this normal case, the FRA has therefore no impact on the scores. More importantly, simulations show that the FRA is able to recover part of the missing detections when some of them were intentionally suppressed. For instance, when 60% of the original detections by a camera are removed, the system can recover up to 40% of the initial HOTA score (without missing detections) of the faulty camera, and 37.5% of the global HOTA score of the crossroads. Nevertheless, not all detections are recovered due to the way the FRA operates. The FRA is based on detections from the previous time frame. If no detections are made, none can be recovered. The FRA is therefore useless when a camera is totally out of service. Nevertheless, an improvement has been proposed to address this issue.
Simulations are also run on a second intersection. Results from these simulations are essentially similar that the ones achieved on the first crossroad, thus strengthening the robustness of the conclusions drawn.

Another objective of the system is that it can be used in real-time. A temporal performance analysis shows that the code can be run in real-time provided the speed at which the images are delivered by a camera is not higher than 40 frames per second (FPS). For the AI city dataset, cameras provide images at the speed of 10 FPS, such that the system can be run in real-time in this instance.

Overall, this document demonstrates the performance of the FRA. It also suggests interesting perspectives to improve the system and identifies other promising applications for the FRA system. Examples include a new FRA activation function or an improvement of the SORT algorithm through feedback.

Appendix A

Link to resources

A.1 Code

The code is available at `https://github.com/pierrecaprasse/fault_tolerant_tracking`.

A.2 Video of the algorithm in action

A one-minute video of the fault-tolerant multi-tracking system with FRA is available at `https://www.youtube.com/watch?v=Qjpm3Hahs4E`.

The parameters used are:

- Faulty camera: 1
- Rate of missing detections: 10%
- FRA: active

Look closely at the detections of camera 1 (top left). The center point of the detection becomes larger when the FRA is activated.

Appendix B

Pseudo Code

The code is divided into three main parts: single-camera tracking, inter-camera tracking and evaluation. The algorithm used for the single-camera tracking step is SORT. The paper corresponds to the implementation used in the article [2]. The code used for the evaluation part is available on the site [12]. Three pseudo-codes running the main parts of the inter-camera tracking are available below:

Algorithm 1 Main loop

```
1: procedure MAIN LOOP
2:     init Clusters
3:     while new frame is available do
4:         for each camera do
5:             load all bounding boxes from SORT algorithm
6:         Clusters ← ICT(Clusters)
7:         if plot then
8:             plot occupancy map
```

Functions where a pseudo code is is not useful for the understanding:

- $AdaptDistances(distances, A, B)$: Will recompute all distances in *distances* knowing that tracker A and B are matched together.
- $number\_common\_trackers(C, OC)$:computes the number of common trackers in cluserts C and OC.
- $on\_image(recovered\_detection)$: will check if the *recovered_detection* is in the field of view of its camera.

Algorithm 2 Inter Camera Tracking

```
1: procedure ICT(oldClusters)
2:     for each tracker A do          ▷ compute distance between trackers
3:         for each tracker B do
4:             if A! = B then
5:                 distances.append(A,B,computeDistance(A,B)
6:
7:     init Cluster                                  ▷ Make the clusters
8:     minDist ← min(distances)
9:     while min(distances) < treshold distance do
10:        trackerA, trackerB ← distances(minDist)
11:        clusters.append(trackerA,trackerB )
12:        distances ← AdaptDistances( distances, A, B)
13:
14:    for C in Clusters do                          ▷ Matching step
15:        for OC in oldClusters do
16:            if number_common_trackers(C, OC) ≥ 2 then
17:                C.global_id ← OC.global_id
18:
19:    for C in Clusters do                          ▷ Feedback step
20:        OC ← oldClusters(where global_id = C.global_id)
21:        interintersection(C, 0C)
22:        if size(inter) < size(OC) then
23:            missingTrackers ← 0C − inter
24:            for MT in missingTrackers do
25:                recovered_detection ← feedback(MT, OC, C)
26:                if on_image(recovered_detection) then
27:                    C.append(recovered_detection)
```

Algorithm 3 Feedback Reconstruction Algorithm

```
1: procedure FEEDBACK(TRACKER,OC,C)
2:     old_other_tracker ← OC.trackers(where id! = tracker.id)
3:     dist ← distance(old_other_tracker, tracker)
4:     new_other_tracker ← C.trackers(where id = old_other_tracker.id)
5:     recovered_tracker = new_other_tracker + dist
```

Appendix C

Additional Results

C.1 Yolov3 detections

An assumption was made that using the detection from the ground truth instead of a detection algorithm such as Yolov3 should not change the fundamental conclusions that could be made during the analysis of the results. Comparing image (a) in figure 4.1 to C.1. The HOTA scores from Yolov3 are lower as expected. Indeed the bounding boxes are less accurate. However, the feedback is still working properly and reconstructing some missing detections.

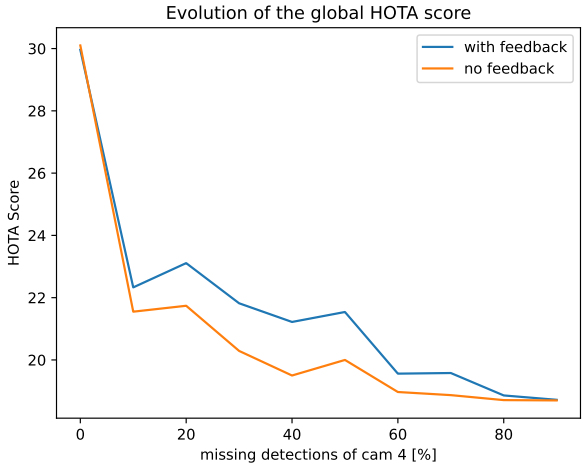


Figure C.1: Evolution of the global HOTA scores depending on the percentage of missing detections by camera 4. Yolov3 detections are used instead of the ground truth detections.

C.2 Faulty camera 3

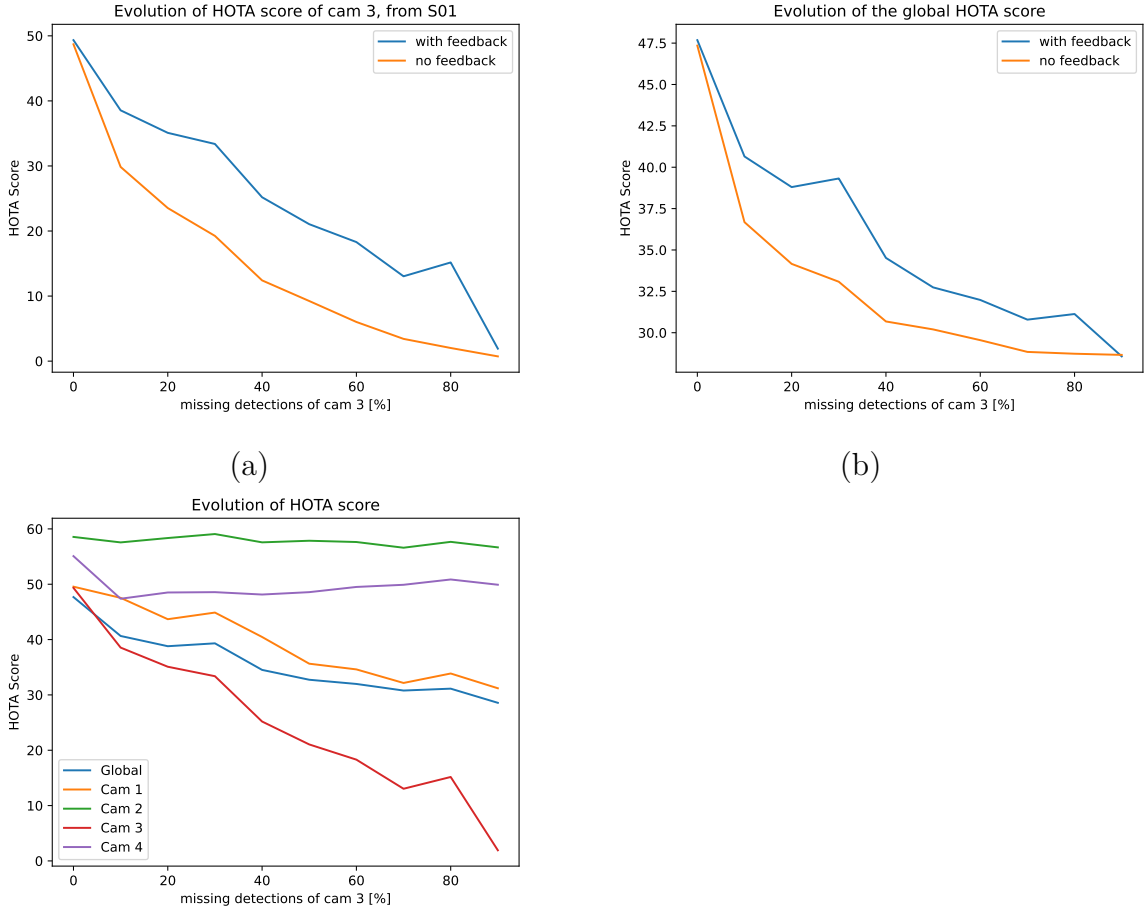


Figure C.2: Evolution of different HOTA scores depending on the percentage of missing detections by camera 3. On (a), the evolution of the score of camera 3 with and without feedback, on (b) the evolution of the global score with and without feedback and on (c) the evolution of all cameras' scores using the feedback.

Bibliography

[1] J. Leclère A. Pisvin. Resilient system for multi-tracking problem using several cameras on raspberry pi, 2022. Prom. : B. Macq, D. Manjah, V. Sonneville.

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[3] Zhenzhong Chen, Weihang Liao, Bin Xu, Hongyi Liu, Qisheng Li, He Li, Chao Xiao, Hang Zhang, Yiming Li, Wentao Bao, and Daiqin Yang. Object Tracking over a Multiple-Camera Network. In *2015 IEEE International Conference on Multimedia Big Data*, pages 276–279, Beijing, China, April 2015. IEEE.

[4] Daniel Pleus. (44) Object Tracking - SORT and DeepSort | LinkedIn. `https://www.linkedin.com/pulse/object-tracking-sort-deepsort-daniel-pleus/`, March 2022. Accessed: 2022-05-2.

[5] Hung-Min Hsu, Jiarui Cai, Yizhou Wang, Jenq-Neng Hwang, and Kwang-Ju Kim. Multi-Target Multi-Camera Tracking of Vehicles Using Metadata-Aided Re-ID and Trajectory-Based Camera Link Model. *IEEE Transactions on Image Processing*, 30:5198–5210, 2021. Conference Name: IEEE Transactions on Image Processing.

[6] Jonathon Luiten. HOTA for Multi Camera · Issue #31 · JonathonLuiten/TrackEval. `https://github.com/JonathonLuiten/TrackEval/issues/31`, April 2021. Accessed: 2022-03-11.

[7] Mahmut Karakaya and Hairong Qi. Collaborative localization in visual sensor networks. *ACM Transactions on Sensor Networks*, 10(2):1–24, January 2014.

[8] Karleigh Moore, Nathan Landman, and Jimin Khim. Hungarian Maximum Matching Algorithm | Brilliant Math & Science Wiki. `https://brilliant.org/wiki/hungarian-matching/`. Accessed: 2022-04-7.

[9] Philipp Kohl, Andreas Specker, Arne Schumann, and Jurgen Beyerer. The MTA Dataset for Multi Target Multi Camera Pedestrian Tracking by Weighted Distance Aggregation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4489–4498, Seattle, WA, USA, June 2020. IEEE.

[10] C. Laoudias, P. Tsangaridis, M. Polycarpou, C. Panayiotou, C. Kyrkou, and T. G. Theocharides. Fault-tolerant tracking of multiple targets in collaborative Camera Networks. In *2015 European Control Conference (ECC)*, pages 3191–3196, Linz, Austria, July 2015. IEEE.

[11] Yen-Cheng Liu, Junjiao Tian, Nathaniel Glaser, and Zsolt Kira. When2com: Multi-Agent Perception via Communication Graph Grouping. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4105–4114, Seattle, WA, USA, June 2020. IEEE.

[12] Jonathon Luiten. TrackEval. `https://github.com/JonathonLuiten/TrackEval`, May 2022. Accessed: 2022-05-16.

[13] Jonathon Luiten, Aljoˇsa Osˇep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. HOTA: A Higher Order Metric for Evaluating Multi-object Tracking. *International Journal of Computer Vision*, 129(2):548–578, February 2021.

[14] MATLAB. Optimal State Estimator Algorithm | Understanding Kalman Filters, Part 4. `https://www.youtube.com/watch?v=VFXf1lIZ3p8`, April 2017. Accessed: 2022-03-11.

[15] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Yue Yao, Liang Zheng, Pranamesh Chakraborty, Christian E. Lopez, Anuj Sharma, Qi Feng, Vitaly Ablavsky, and Stan Sclaroff. The 5th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021.

[16] Opencv. Basic concepts of the homography explained with code. `https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html`. Accessed: 2022-05-11.

[17] Alejandro Suarez, Guillermo Heredia, and Anibal Ollero. Cooperative Virtual Sensor for Fault Detection and Identification in Multi-UAV Applications. *Journal of Sensors*, 2018:1–19, 2018.

[18] Minghu Wu, Yeqiang Qian, Chunxiang Wang, and Ming Yang. A Multi-Camera Vehicle Tracking System based on City-Scale Vehicle Re-ID and Spatial-Temporal Information. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4072–4081, Nashville, TN, USA, June 2021. IEEE.

[19] XTER.FR. Qu'est-ce qu'une ville connectée ? `https://www.xter.fr/quest-ce-quune-ville-connectee/`, 2022. Accessed: 2022-04-2.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**