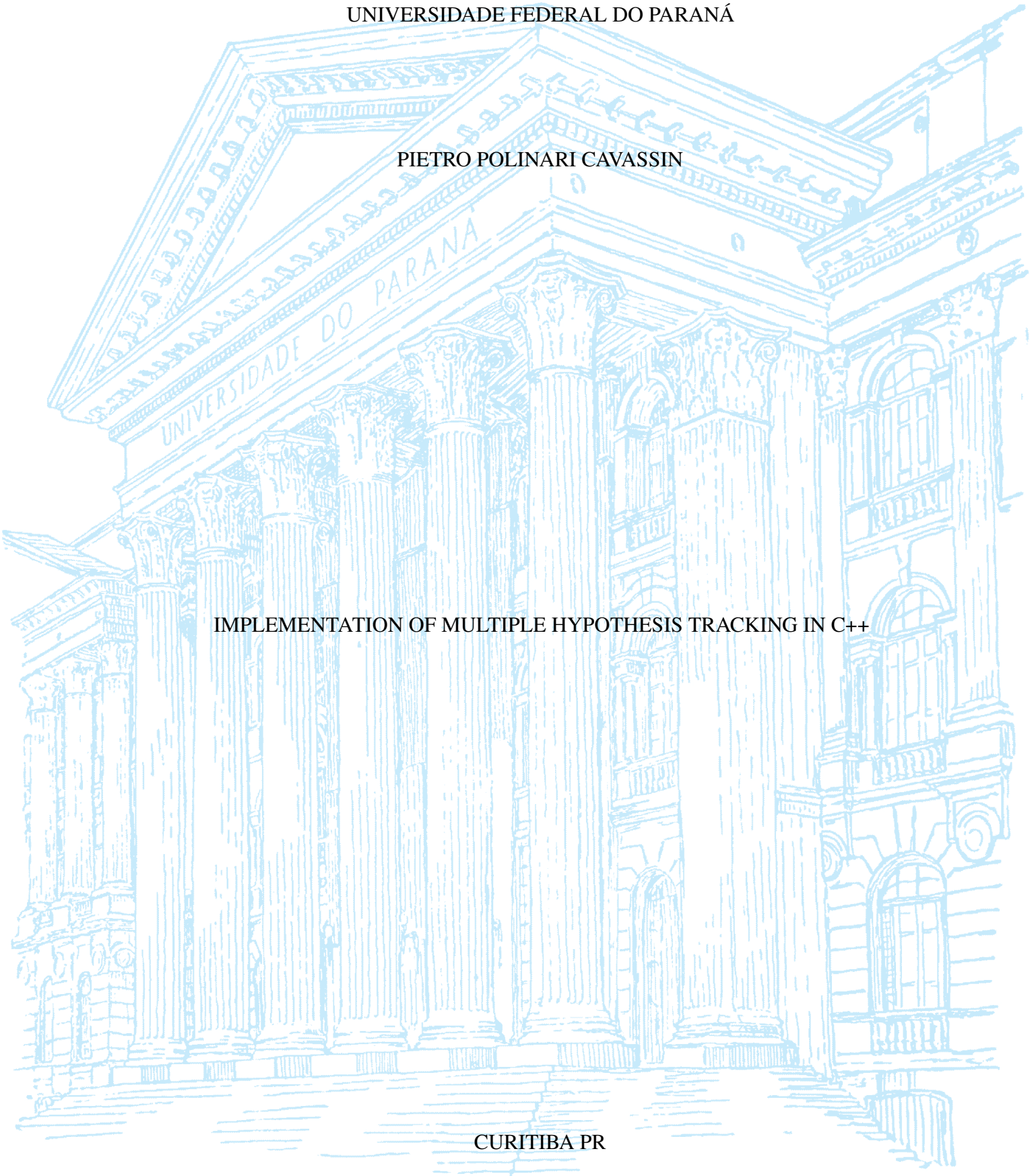
UNIVERSIDADE FEDERAL DO PARANÁ

PIETRO POLINARI CAVASSIN



IMPLEMENTATION OF MULTIPLE HYPOTHESIS TRACKING IN C++

CURITIBA PR

2024

PIETRO POLINARI CAVASSIN

IMPLEMENTATION OF MULTIPLE HYPOTHESIS TRACKING IN C++

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Computação*.

Orientador: Fotios Katsilieris, Eduardo Todt.

CURITIBA PR

2024

To my parents, who always praised my growth and well-being and, above all else, gave me all the love I needed.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to all those who have contributed to the completion of this thesis.

Firstly, I would like to thank the AWARE team at Technische Hochschule Ingolstadt. Their invaluable support and guidance facilitated my internship at Airbus Defence and Space in Ingolstadt. Without their efforts, this opportunity would not have been possible. I am also very grateful for their help in navigating the challenges of living and working in a foreign country.

I am also thankful for Airbus Defence and Space in Manching for providing me with a scholarship to undertake this internship. Their generous support made it possible for me to focus on my research and gain invaluable experience.

I would like to extend my thanks to my advisor at Airbus, Fotios Katsilieris, for his unwavering support and guidance. His expertise, patience, and willingness to share knowledge have significantly enriched my research experience.

Finally, I would like to express my deepest gratitude to my family for their love, support, and encouragement. Their belief in me and their constant support were the foundation that brought me to this journey and kept me persistent throughout it.

RESUMO

Este trabalho aborda os desafios do Rastreamento de Múltiplos Objetos (RMO) com a implementação do algoritmo de Rastreamento de Múltiplas Hipóteses Orientado a Trajetórias (RMHOT) e a integração de métodos de estimativa Bayesiana dentro de um framework do Filtro de Kalman. A motivação do trabalho vem da crescente necessidade de rastreamento em tempo real em sistemas dinâmicos, onde a precisão das trajetórias de múltiplos objetos é essencial. Este trabalho envolve o estudo, desenvolvimento e avaliação de um rastreador funcional e genérico que utiliza a técnica de RMHOT para resolver o problema de RMO e que pode servir como base para melhorias e otimizações futuras. As técnicas de Gating e poda N-scan foram utilizadas para reduzir o número de trajetórias candidatas. A associação de dados é feita em uma janela de N-scan de medições, sendo reduzida ao problema de Subgrafo Independente de Peso Mínimo, que é equivalente ao problema de atribuição S-dimensional. A solução para esse problema foi feita de maneira ingênua com enumeração de subconjuntos, tornando-se o gargalo do algoritmo implementado. O rastreador foi bem-sucedido em rastrear objetos na presença de falsos alarmes e detecções perdidas, embora o tempo de processamento cresça rapidamente com o número de detecções e janelas de varreduras devido ao tempo necessário para realizar a associação de dados. Outro problema nos resultados do algoritmo foi que os falsos alarmes não foram descartados, mas considerados como um objeto separado. Apesar das limitações, esta tese resultou em um ponto de partida sólido para a implementação de um rastreador RMHOT que pode ser aprimorado no futuro.

Palavras-chave: Rastreamento de Múltiplos Objetos. Rastreamento de Múltiplas Hipóteses. Implementação.

ABSTRACT

This thesis addresses the challenges of MOT (Multiple Object Tracking) with an implementation of the TO-MHT (Track-Oriented Multiple Hypothesis Tracking) algorithm and the integration of Bayesian estimation methods within a Kalman Filter framework. The motivation of the thesis comes from the growing need of real-time tracking in dynamic systems, where the accurate of the trajectories of multiple objects is essential. This work involves the study, development and evaluation of a functional and generic tracker that makes use of TO-MHT to solve the MOT problem and can serve as a basis for future improvements and optimizations. Gating and N-scan pruning were the techniques used to reduce the number of candidate tracks. The data association is made in an N-scan window of measurements, being reduced to the Minimum Weight Independent Subgraph problem, which is equivalent to the S-dimensional assignment problem. The solution for this problem was made in a naïve way with subset enumeration, becoming the bottleneck of the implemented algorithm. The tracker was successful to track objects in the presence of false alarms and missed detections, although the processing time grows fast with the number of detections and scans due to the time taken to make the data association. Another problem in algorithm results was that false alarms were not discarded, but were rather considered as a separate track. Despite the drawbacks, this thesis resulted in a solid starting point for the implementation of a TO-MHT tracker that can be further enhanced.

Keywords: Multiple Object Tracking. Multiple Hypotheis Tracking. Implementation.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

BGH	Best Global Hypothesis
DAP	Data Association Problem
EAP	Expected a Posteriori
GNN	Global Nearest Neighbor
JPDA	Joint Probabilistic Data Association
LLR	Log-Likelihood Ratio
MHT	Multiple Hypothesis Tracking
MMSE	Minimum Mean Squared Error
MOT	Multiple Object Tracking
MWIS	Maximum Weighted Independent Set
NLLR	Negative Log-Likelihood Ratio
PDF	probability density function
PMF	probability mass function
TO-MHT	Track-Oriented Multiple Hypothesis Tracking

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

The field of MOT (Multiple Object Tracking) consists on sequentially processing noisy sensor data to determine the number of moving objects and estimating their states and trajectories in space (Svensson and Granström, 2019).

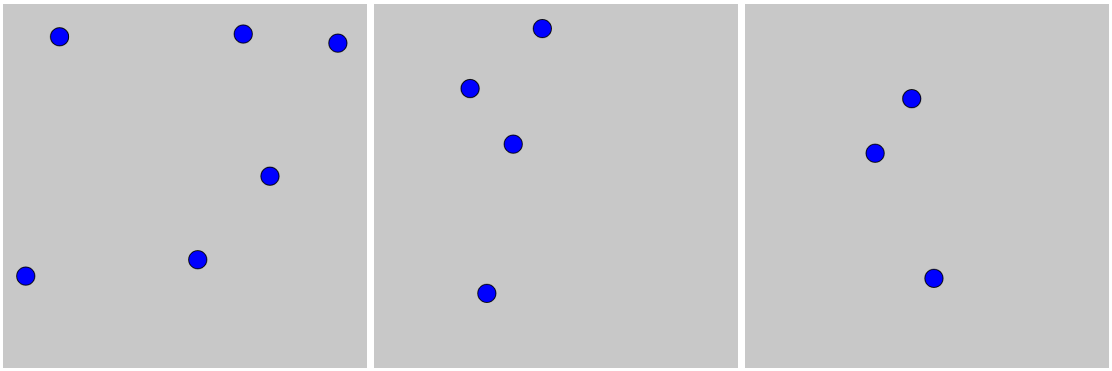


Figure 1.1: Sequence of consecutive sensor measurements where each circle corresponds to a detection that can either be object-originated or clutter.

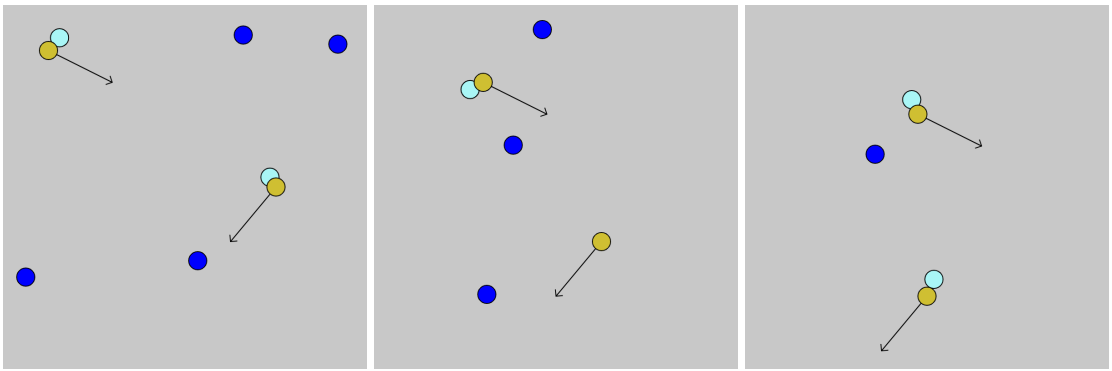


Figure 1.2: Same Multiple Object Tracking example as Figure 1.1, where object-originated mesaurements are highlighted in light blue, while the clutter remains colored in dark-blue. In addition, the yellow circles correspond to the actual states of the moving objects and the arrows correspond to their velocities. Note that the lower object was not detected in the middle frame.

Tracking objects has a wide range of applications. One of the earliest historically registered uses of object tracking dates back to the decade of 1930, with the ascending interest for detecting invading aircrafts in many countries' airspaces due to the international diplomatic tensions that led to the Second World War. To this day, this application is used in redundancy with other methods in the tracking of modern aircrafts. Another widely-used application of Multiple Object Tracking is for autonomous vehicles. Besides the need of mechanisms to recognize the static environment around them, it is also essential that they are able to identify and track other moving objects, such as pedestrians, cyclists and other vehicles. This awareness allows autonomous cars prevent potential dangerous situations and make quick decisions to avoid accidents in case of hazard. Among the other possibilities are the tracking groups of pedestrians to plan emergency procedures, tracking the movements of cells for microfluidic studies, ground

radar tracking of airplanes in airports as a useful redundancy method in poor weather conditions and tracking the movement of sports players in a field for the collection of statistics. Overall, the usage of object tracking revolves around understanding the trajectories of individuals and/or groups in their environments from sensor detections.

In the general case, the inputs for MOT algorithms are sets of measurements of detected objects over time, as illustrated in Figure 1.1. With that data, the algorithm is expected to identify how many moving objects are being detected and which measurements correspond to which objects at different moments of time. Based on that, the goal is to identify their trajectories, as shown in Figure 1.2. The problem of associating the sensor measurements with their corresponding real-life objects is called the **Data Association Problem**. In order to solve it, a set of different obstacles need to be overcome (Svensson and Granström, 2019).

At first, the origin of the measurements is unknown, that is, there is no information about which measurement corresponds to each object. It is also not known whether all the measurements correspond to objects because sensor data usually comes with noise, and this originates sets of detections that do not correspond to objects of interest, also known as clutter. The unknown origin of the measurements and the clutter are illustrated in Figure 1.1.

The occurrence of missed detections — exemplified in the middle frame of Figure 1.2 — is also common due to sensor failures or obstacles blocking object detection. Therefore, the algorithms need mechanisms to fill these gaps and identify the correspondence between non-consecutive measurements originated by the same targets.

In addition, no sensor is perfectly precise, so noisy measurements are unavoidable. This means that every object detection also includes some uncertainty of the its real position.

Finally, a small distance between two targets can also be a source of errors due to the limited measurement resolution from sensors. The closer two objects are, the higher is the chance that a sensor produces only one measurement for both. Even when close objects are detected separately, it is still hard for tracking algorithms to differentiate which detection corresponds to each object.

These challenges are addressed in different ways by different Multiple Object Tracking algorithms. One of the most popular approaches to Multiple Object Tracking is the Bayesian Approach, whose main feature is called **Bayesian Filtering** and involves two main steps that are repeated after each measurement is made by the sensor: **prediction** and **update**.

The prediction step requires a **motion model**, which describes the motion of an object and its uncertainty based on its current state. The motion model can change depending on the object, since different objects move in different ways. The prediction uses the current state and motion model of an object to estimate its state and uncertainty in the next scan. Once the next scan is done and a measurement is associated with the track object, comes the update step. In the update step, the previous prediction and the associated measurement, together with a **measurement model** are used to update the probability density of the object's true state. The measurement model relates the object's state to its measurement and uncertainty. Measurement models can vary between sensors and detected objects, given their different natures. The resulting state of the update is then used by the next prediction step and the steps are recursively alternated, as shown in Figure 1.3.

The Data Association Problem can be tackled using various approaches that differ in their ways of dealing with the exponentially growing number of data association hypotheses. Some examples are GNN (Global Nearest Neighbor), JPDA (Joint Probabilistic Data Association) and MHT (Multiple Hypothesis Tracking).

The basic idea of the GNN approach is to, in each update, greedily select the most likely data association and prune all other possible associations (Bar-Shalom et al., 2011). This

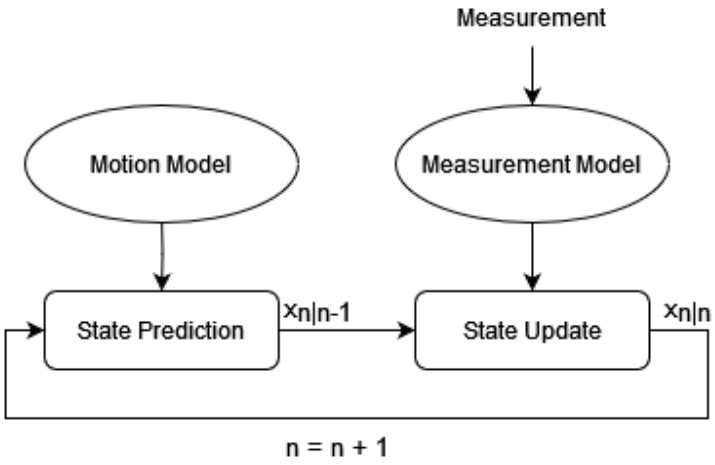


Figure 1.3: Flowchart of Bayesian Filtering where n corresponds to the current timestamp, $x_{n|n-1}$ is the predicted state of x at time n based on its previous state at time $n-1$ and $x_{n|n}$ is the updated state of x at time n based on the measurement made at time n.

means that only the best association is considered at each time step and all others are ignored. On the one hand, this approach has simple implementation, is computationally cheap and works well with well-behaviored sensor data. On the other hand, selecting the most likely associations at each timestamp does not guarantee that the hypotesis (sequence of data associations since the beginning) is the most likely. In addition, in the cases where there is a higher presence of clutter and lower measurement certainty or when two or more tracked objects are close together, the tracking performance of the algorithm can drastically decrease (Svensson and Granström, 2019). Overall, the main obstacle for GNN is the presence of many measurements that have a considerable likelihood of being associated to the same object.

In opposition to GNN, the JPDA algorithm considers many possible data associations for each object at each time step, instead of only the most likely (Bar-Shalom et al., 2011). The nearest neighbors are assigned higher weights while the farthest are given lower weights, and the mean and covariance in the update step are calculated based on the measurements and their respective weights. Its main advantages over GNN are that it has better performance on cases with more clutter and uncertainty, it is computationally cheap and relatively simple to implement. On the downside, according to Svensson and Granström (2019), its performance significantly decreases on scenarios where objects go close to each other, since it is likely that the measurements of more than one object will be considered for each of the objects' state estimate.

Both the previous algorithms consider only one hypothesis, with the difference that JPDA takes advantage of information from more than one measurement, while GNN loses information from all the pruned measurements. On the other hand, Multiple Hypothesis Tracking, as its name suggests, considers many hypotheses over several time steps (Svensson and Granström, 2019). In the update step, the most likely data associations for each object originate a number of new hypotheses to be considered by the algorithm. Afterwards is added a new third **reduction** step, where some measures to reduce memory consumption are taken. For example, old data points are deleted and the number of considered hypotheses is reduced below a defined maximum in a way that only the most likely hypotheses are kept. The benefit of this approach is that considering many hypotheses with high likelihood results in a higher chance of keeping hypotheses with many correct data associations. Besides that, the limited number of considered hypotheses allows the computational cost of the algorithm to be controlled.

The MHT algorithm has two main variants: the simpler and more naïve **hypothesis-oriented** approach and the more memory-efficient **track-oriented** one. The difference between them lies on the way that hypotheses are stored, where the former explicitly stores each of the currently considered data association hypotheses, while the latter separately stores the data association hypotheses for each target (local hypotheses) and builds a global hypothesis out from the local ones (Bar-Shalom et al., 2011). The advantage of the second approach over the first one is that in a much lower amount of space it is possible to store a much larger number of data association hypotheses, as they are implicitly stored. The first approach leads to multiple copies of the same track in different hypotheses.

1.2 GOALS OF THESIS

The goal of this thesis is straightforward: to understand, explain and implement the algorithm of TO-MHT (Track-Oriented Multiple Hypothesis Tracking). The algorithm will be implemented in a generic way, so it can be applied in many different occasions, not depending on the type of object being tracked, nor on the way sensor data is retrieved.

1.3 STRUCTURE OF THESIS

This thesis explains the mathematical fundamentals behind Multiple Hypothesis Tracking in Chapter 2, ensuring a foundation for the understanding of the theory behind the techniques used in the algorithm's implementation. In Chapter 3, some of the best performing state-of-the-art algorithms for Multiple Object Tracking are explained. Chapter 4 showcases how the algorithm was implemented with pseudo-codes and the main data structures that were developed and used in it. Afterwards, Chapter 5 explains how the algorithm was tested, while Chapter 6 shows the results achieved and discusses what was expected, what could be improved and how it could be improved. To summarize this thesis, Chapter 7 goes back to what was proposed and what was achieved, further discussing what a future work could do to improve the performance of the implementation.

2 THEORETICAL FUNDAMENTALS

In the last chapter, we had a general view of the Multiple Object Tracking problem and the obstacles that come with it. In this chapter, we will have a more in-depth view of all the tools that are used to come over these obstacles, starting with Estimation Theory basics and concluding at the Track-Oriented Multiple Hypothesis Tracking algorithm.

2.1 ESTIMATION THEORY AND BAYESIAN ESTIMATION

2.1.1 Estimation in static systems

According to Bar-Shalom et al. (2001), the term parameter designates a time-invariant quantity. The problem of estimating a parameter x is to, given various measurements z_j, $j \in [1..n]$ made in presence of noise w_j, find a function $\hat{x}(O)$ that estimates the value of x, given the set of measurements $O = \{z_j\}_{j=1}^{n}$.

The inherent uncertainty associated with real-world sensor data makes the use of **estimation theory** essential for tracking the states of detected objects. The Multiple Hypothesis Tracking approach for solving the Multiple Object Tracking problem makes the use of **Bayesian estimation**. This subsection will be dedicated to reviewing the principles of estimation theory and Bayesian estimation.

Estimation theory is a branch of statistics dedicated to estimating the values of unknown parameters of interest using observed data. Let the vector $\mathbf{x} \in \mathcal{X}$ be the state or parameter and the vector $\mathbf{z} \in \mathcal{Z}$ be the measurement. The probability distribution of the observation $\mathbf{z}$ given the object state $\mathbf{x}$ is represented by the *likelihood* function $p(\mathbf{z}|\mathbf{x})$. Therefore, in our case, the likelihood function represents where it is likely that the sensor will detect the object, given its actual position.

The Bayesian paradigm of estimation theory requires the *prior* probability distribution of the object state $\mathbf{x}$, which is given by the probability density function $p(\mathbf{x})$. In our case, the prior probability represents where it is likely that the actual position of the object is before it is measured.

Given the prior and the likelihood PDFs, the Bayes rule is used to compute the *posterior* function $p(\mathbf{x}|\mathbf{z})$

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{\int p(\mathbf{z}|\mathbf{x})p(\mathbf{x})d\mathbf{x}} \tag{2.1}$$

, which is the probability distribution of the state of the detected object (parameter) given its observation. For us, the posterior probability distribution represents where it is likely that the object is, based on its measurement.

Although the PDF $p(\mathbf{x}|\mathbf{z})$ is descriptive of what the parameters are likely to be, it does not stipulate a value for them. This role is performed by the **estimator**, which is a function $\hat{\mathbf{x}}$ that assigns an observation $\mathbf{z}$ to a value $\hat{\mathbf{x}}(\mathbf{z}) \in \mathcal{X}$. To evaluate the performance of an estimation made by an estimator function, a cost function $C(\hat{\mathbf{x}}(\mathbf{z}), \mathbf{x})$ is used. A **Bayes optimal estimator** is a function that minimizes the Bayes Risk, which is the expected cost over all the realizations of observation and state (ngu Vo et al., 2015).

One commonly used estimator is the EAP (Expected a Posteriori)

$$\hat{\mathbf{x}}_{\mathrm{EAP}}(\mathbf{z}) = \int_{-\infty}^{\infty} \mathbf{x} p(\mathbf{x}|\mathbf{z})\, d\mathbf{x}. \tag{2.2}$$

The EAP estimator is the Bayes optimal estimator for the cost function $C(\hat{\mathbf{x}}(\mathbf{z}), \mathbf{x}) = \|\hat{\mathbf{x}}(\mathbf{z}) - \mathbf{x}\|^2$. In other words, the EAP estimator is the Minimum Mean Squared Error estimate for the object's state (ngu Vo et al., 2015).

2.2 THE KALMAN FILTER

In the previous subsection, estimation of static variables was described, but in Multiple Object Tracking, the states of the objects are dynamic as they move over time. Therefore, dynamic estimation must be explained.

2.2.1 Bayes Recursion

In estimation of dynamic systems, time can be dealt with in two different ways: continuous and discrete. In this thesis, we deal with time in the discrete form. In this form, at each time step k, the observed object is in a state $\mathbf{x}_k$ and generates a measurement $\mathbf{z}_k$. In dynamic estimation, we not only use the current measurement of an object to estimate its position, but we also use all its measurement history.

A way of doing the estimation of dynamic systems using the Bayes rule is the **Bayes recursion**, which consists of two main steps: **filtering** and **update**. In the filtering step is calculated the **filtering density** $p_k(\mathbf{x}_k|\mathbf{z}^k)$, which is the probability density of an object's state at time step k given its measurement history from time step 1 to k, $\mathbf{z}^k$. In the filtering step is calculated the **prediction density** $p_{k+1|k}(\mathbf{x}_{k+1}|\mathbf{z}^k)$, which is the probability density of an object's state at time step $k+1$ given its measurement history up until time step k (ngu Vo et al., 2015).

The first filtering is done using a prior PDF for the object and the first measurement. Then, the prediction step uses the first filtering density to predict where the next measurement will be. The second filtering step uses the prediction made in the previous step as a prior PDF, and the recursion continues.

The Kalman Filter is an estimator that makes use of the Bayes recursion assuming that the initial state and all the noises in the system are Gaussian. Under those conditions, it is the optimal MMSE estimator for the object's state (Bar-Shalom et al., 2001).

The Gaussian assumption allows all the states and measurements in the Kalman Filter be described by a mean vector and a covariance matrix. For each time step k, there exists a state prediction $\hat{x}_{k|k-1}$ and its corresponding covariance matrix $P_{k|k-1}$.

2.2.2 The Kalman Update

At a given time step k, in the **update** phase of the Kalman Filter, the measurement z_k is used to update the values of the state $\hat{x}$ and its covariance P.

This phase requires the use of a **measurement model**. The measurement model describes the measurement made by the sensor based on the object's state. Let x be the state vector, corresponding to the objects's cartesian coordinates x and y and the rates of change $\dot{x}$ and $\dot{y}$, i.e.

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}$$

and $\mathbf{z}$ be the vector that represents the measurement made by the sensor, corresponding only to the object's cartesian coordinates x and y, i.e.

$$\mathbf{z} = \begin{bmatrix} x \\ y \end{bmatrix}$$

.

The measurement model matrix H would be the matrix such that, when it multiplies the state vector x, results in the measurement $\mathbf{z}$, i.e. $H\mathbf{x} = \mathbf{z}$. In this example,

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

.

The measurement model should also consider noise, i.e. uncertainty in the precision of the measurement. This uncertainty is represented by the covariance matrix R, which represents a Gaussian noise $v \sim \mathcal{N}(v; 0, R)$.

At a given time step k, the Kalman update receives as input the newly detected measurement z_k. It then uses the measurement model matrix H and the measurement noise covariance R in conjunction with the new measurement previously predicted state $\hat{x}_{k|k-1}$ and covariance $P_{k|k-1}$ to update the estimations for the object's state and covariance. The computation of these variables use three intermediate values: The **innovation** ν_k, the **innovation covariance** S_k and the **filter gain** W_k

$$\nu_k = z_k - H_k \hat{x}_{k|k-1} \tag{2.3}$$
$$S_k = R_k + H_k P_{k|k-1} H_k^T \tag{2.4}$$
$$W_k = P_{k|k-1} H_k^T S_k^{-1} \tag{2.5}$$

, where

- H_k is the measurement model matrix,
- R_k is the measurement noise covariance.

The innovation ν_k captures the new information given by the most recent measurement $\mathbf{z}_k$. It is the difference between the new measurement $\mathbf{z}_k$ and the measurement prediction $\hat{\mathbf{z}}_{k|k-1} = H_k \hat{\mathbf{x}}_{k|k-1}$.

The filter gain intuitively indicates the scale of the response of the state update to the measurement, where a larger filter gain indicates a higher response to the measurement and a lower filter gain corresponds to a lower response to the measurement. Taking a simplistic scalar view of the filter gain, the more "inaccurate" the state prediction and the more "accurate" the measurement are, the higher is the filter gain (Bar-Shalom et al., 2001).

Finally, the state estimate is updated using the measurement prediction $\hat{\mathbf{x}}_{k|k-1}$, the filter gain W_k and the residual ν_k. The covariance is updated using the predicted covariance $P_{k|k-1}$, innovation covariance S_k and filter gain W_k

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + W_k \nu_k \tag{2.6}$$
$$P_{k|k} = P_{k|k-1} - W_k S_k W_k^T. \tag{2.7}$$

2.2.3 The Kalman Prediction

In the **prediction** phase of the Kalman Filter at time step k, the predictions for the next state $\hat{\mathbf{x}}_{k+1|k}$ and its covariance matrix $P_{k+1|k}$ are made. To make these predictions, the Kalman Filter needs a **/motion model** to describe how the state of the object changes over time.

A simple motion model is the model of constant velocity. In this case, our motion model matrix F_k would be

$$F_k = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

such that, when it is multiplied by the state of the object at a given time step, it results on its next state considering a motion in constant velocity, i.e.

$$\begin{aligned} F_k x_k &= \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} \\ &= \begin{bmatrix} x + \dot{x}T \\ \dot{x} \\ y + \dot{y}T \\ \dot{y} \end{bmatrix}. \end{aligned}$$

It is clear that the velocities on each axis remain constant while the coordinates change according to the velocities.

The motion model also includes the process noise covariance matrix Q_k, which describes the Gaussian noise in the changing of states of the object, e.g. changes on its speed and position over time.

Both F_k and Q_k matrices are used with the updated state and covariance matrix at time step k to predict the state and covariance matrix at the next time step $k + 1$:

$$\hat{\mathbf{x}}_{k+1|k} = F_k \hat{\mathbf{x}}_{k|k} \tag{2.8}$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k. \tag{2.9}$$

The mathematical derivation of the Kalman Filter equations can be consulted in Bar-Shalom et al. (2001).

2.3 MULTIPLE OBJECT TRACKING AND MULTIPLE HYPOTHESIS TRACKING

Up until the previous section, we described an effective way of estimating the trajectory of a moving object over time, assuming that its measurements at each time step are known. In this section, we will describe the principles of tracking multiple moving objects, the new problems that arise with it and how the MHT algorithm overcomes these obstacles.

2.3.1 Multiple Object Tracking Systems

In Multiple Object Tracking, at each time step k, a set of measurements Z_k , with cardinality m_k, is detected by a sensor.

$$Z_k = \{\mathbf{z}_k^1, \mathbf{z}_k^2, \cdots, \mathbf{z}_k^{m_k}\}$$

, where $\mathbf{z}_k^i$ corresponds to measurement i at time step k. Nevertheless, this input does not accurately represent the real-world scenario. As previously explained, to implement a Multiple Object Tracking algorithm, the following recurring problems need to be solved.

- **Sensor Noise**: Sensors have imperfections, so there is always noise in the measurements. Therefore, the detections never correspond to the exact states of the objects.
- **Data Association Ambiguity**: It is not known which measurements correspond to which objects.
- **Missed Detections**: Occasional failures on the sensors or the presence of physical obstacles can cause some objects to be occluded and consequently not detected.
- **False Alarms**: Irrelevant objects, clutter and sensor failures can generate measurements that do not correspond to objects of interest (false alarms).
- **Variable Number of Targets**: As targets move over time, they can exit and enter the scanned area, and the algorithm has to identify these variations.

As we will see in the next subsection, the MHT algorithm tries to keep a set of hypotheses for the association between objects and measurements. At each time step, the algorithm receives a new set of measurements and uses them to propagate the existing hypotheses into new ones. Afterwards, it calculates the posterior probabilities of the new hypotheses using the Bayesian Estimation and maintains the most likely ones. With time, the number of hypotheses can grow exponentially, and the algorithm makes use of pruning techniques to reduce them to an amount that can be computed by real-time applications.

2.3.2 Multiple Hypothesis Tracking

According to Bar-Shalom et al. (2011), Multiple Hypothesis Tracking is a measurement-oriented approach, in the sense that it is evaluated whether a measurement was originated from an established target or a new target. Other approaches, such as GNN and JPDA, are target-oriented, which means that, given a target, the probabilities of each measurement of being originated by it are evaluated. The measurement-oriented approach has the advantage of the possibility of track initiation over the other aforementioned techniques.

To understand the MHT algorithm, we first need to understand what a hypothesis is. A hypothesis is a sequence of associations for each of the measurements made at each time step k. For each measurement, there are three feasible associations (Bar-Shalom et al., 2011):

1. The measurement is the continuation of an existing track,
2. The measurement corresponds to a new track,
3. The measurement is a false alarm.

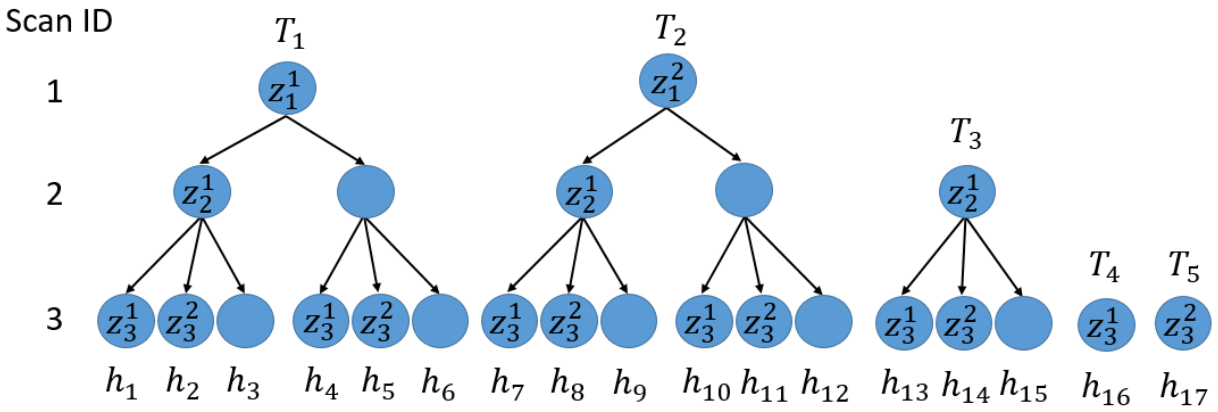


Figure 2.1: Illustration of the track trees generated by the measurements at equations 2.10 to 2.12 without any gating and pruning techniques. Over the root node of each of the tree is its identification (e.g. T_1, T_2, ...). Each node of a track tree is represented with the measurement that gave origin to it, or nothing in case it is a missed detection. Each of the leaves of the track trees is a local hypothesis that corresponds to the path from the root of the tree to the leaf. Each level of the tree corresponds to a scan.

In the Track-Oriented Multiple Hypothesis Tracking approach, these possibilities are used to build track trees that represent the hypotheses for the trajectories of the objects. Let us take a simple example for explanation. Suppose that at three consecutive time steps, the following sets of measurements are made:

$$Z_1 = \{\mathbf{z}_1^1, \mathbf{z}_1^2\} \tag{2.10}$$

$$Z_2 = \{\mathbf{z}_2^1\} \tag{2.11}$$

$$Z_3 = \{\mathbf{z}_3^1, \mathbf{z}_3^2\}. \tag{2.12}$$

The implementations of Multiple Hypothesis Tracking contain many optimizations that will be further explained in this thesis. For the sake of this example, these techniques will be currently ignored. On the course of the next subsections, they will be progressively explained.

In our example, as scan Z_1 is the first one and there are no existing tracks, two track initiation hypotheses are created: measurement $\mathbf{z}_1^1$ gives birth to track tree T_1 and $\mathbf{z}_1^2$ to track T_2. In the track tree representation, this is portrayed by two new track roots, as seen in Figure 2.1 . Afterwards, when scan Z_2 is made, the pre-existing tracks are propagated and new ones are created. Measurement $\mathbf{z}_2^1$ can be the continuation of track T_1 or T_2, or even start a new track hypothesis T_3. The missed detection hypothesis should also be created for each track to deal, for example, with the case that measurement $\mathbf{z}_2^1$ initiates a new track. The same approach is used for scan Z_3, creating the set of hypotheses shown in Figure 2.1.

It is easy to see that the number of hypotheses has the potential to grow rapidly. The measures used to mitigate this will be described later in this section.

At a given scan, each of tree leaves is called a **local hypothesis** or **track hypothesis**. It is the hypothesis that the track that starts at the root follows the trajectory given by the measurements (or missed detections) given by its path to the leaf of the tree. For example, in Figure 2.1, the local hypothesis h_4 considers that its trajectory follows measurement $\mathbf{z}_1^1$, a missed detection and measurement $\mathbf{z}_3^2$.

A **global hypothesis** is a set of compatible local hypotheses. Local hypotheses are compatible if they do not have measurements in common. A global hypothesis should also include all the measurements made in the previous N-scan window, which includes all the scans made in the previous N time steps and the current scan. This number N is the parameter for the

size of the N-scan pruning window, which will be further explained at subsection 2.3.6. Global hypotheses are hypotheses for the states of the multiple targets.

As it will be explained in the next subsection, each the local hypotheses will be assigned scores according to their likelihoods. These track scores will be combined into global hypothesis scores, which will be the metric used to calculate the most likely hypothesis for the trajectories of the tracked objects at each scan.

2.3.3 Log-Likelihood Ratio

The LLR (Log-Likelihood Ratio) is a way of evaluating the likelihood of a track, which is used to calculate the likelihood of a hypothesis. There are three different cases of LLR calculation: the likelihood ratio initialization, the continuation likelihood ratio, the no-continuation likelihood ratio.

The appearances of new targets and false alarms are modeled as the following Poisson probability mass functions (Bar-Shalom et al., 2007).

$$p_\phi(\phi) = e^{-\lambda_\phi V}\frac{(\lambda_\phi V)^\phi}{\phi!}$$
$$p_\nu(\nu) = e^{-\lambda_\nu V}\frac{(\lambda_\nu V)^\nu}{\nu!},$$

where

- ϕ: The number of measurements considered **false alarms** in the current hypothesis at the current time step.
- ν: The number of measurements considered **new targets** in the current hypothesis at the current time step.
- V: The surveillance volume, which is the volume of the measurement space.
- λ_ϕ: Expected number of **false alarms** per unit of volume of the measurement space per scan.
- λ_ν: Expected number of **new targets** per unit of volume of the measurement space per scan.

Both λ_ν and λ_ϕ are important parameters that influence the calculation of the track costs. For example, the likelihood ratio of a **new track** is entirely defined by them and is obtained by the following equation (Bar-Shalom et al., 2007).

$$\mathcal{L}_{new} = \frac{\lambda_\nu}{\lambda_\phi}. \tag{2.13}$$

The likelihood ratio of a measurement j being the **continuation of track** t is given by

$$\mathcal{L}_{tj}(k) \triangleq \frac{\Lambda_{tj}(k)}{\Lambda_{0j}(k)} = \frac{f_t[z_j(k)]P_{D_t}(k)}{\lambda_{ex}} \tag{2.14}$$

, where $\Lambda_{tj}(k)$ is the likelihood of measurement j be the continuation of track t, $\Lambda_{0j}(k)$ is the likelihood of it being originated from an extraneous source and $\lambda_{ex} = \lambda_\phi + \lambda_\nu$ is the spacial density of the objects that originate from extraneous sources.

When the track is not associated with any measurement (i.e. when there is a **missed detection**), the likelihood ratio is denoted by

$$\mathcal{L}_{t0} \triangleq \frac{\Lambda_{t0}(k)}{\Lambda_{00}(k)} = \frac{1 - P_{D_t}(k)}{1} = 1 - P_{D_t}(k). \tag{2.15}$$

, where $\Lambda_{t0}(k)$ is the non-continuation likelihood of track t and $\Lambda_{00}(k)$ is when no track is not continued, which does not have any effect in the likelihood and, therefore is set to 1.

Since the previous operations can result in values of orders of magnitude that are not well represented by floating point variables in computers, the logs of these likelihood ratios are calculated. Besides that, the best assignment of measurements to tracks should be found by minimizing a cost function. In our case, the negative of the log likelihood ratio NLLR is used as such a function.

$$\mathsf{NLLR}_t^k \triangleq -\ln \mathcal{L}_t^k = -\sum_{l=0}^{k} \ln \mathcal{L}_{t,j(t,l)}(l) = \sum_{l=0}^{k} \mathsf{NLLR}_{t,j(t,l)}(l) \tag{2.16}$$

If it is also assumed that the Kalman Filter is used, the NLLR for **track continuation** is (Bar-Shalom et al., 2007)

$$\begin{aligned} \mathsf{NLLR}_{tj}(k) &\triangleq -\ln \mathcal{L}_{tj}(k) && (2.17)\\ &= \frac{1}{2}[z_j(k) - \hat{z}(k|k-1)]' S_{tj}(k)^{-1} [z_j(k) - \hat{z}(k|k-1)] \\ &\quad + \ln \frac{\lambda_{ex} |2\pi S_{tj}(k)|^{\frac{1}{2}}}{P_{D_t}(k)} && (2.18) \end{aligned}$$

For **track initiation** and **missed detections**, the calculations of the NLLR are straightforward and do not depend on variables produced by the Kalman Filter.

$$\mathsf{NLLR}_{new} = -\ln \frac{\lambda_\nu}{\lambda_\phi} \tag{2.19}$$

$$\mathsf{NLLR}_{t0}(k) = -\ln \mathcal{L}_{t0}(k) = -\ln[1 - P_{D_t}(k)]. \tag{2.20}$$

The cumulative NLLR of a track t through time k is the sum of the NLLRs calculated at each time step.

$$\ell_t^k = \sum_{l=0}^{k} \mathsf{NLLR}_{tj}(l). \tag{2.21}$$

As will be shown in the next subsection, the NLLRs of the tracks are then used to calculate the best global hypothesis and give an estimation of the actual states of the objects at each time step.

2.3.4 Best Global Hypothesis

Once the NLLRs, or costs of the tracks have been calculated, the BGH (Best Global Hypothesis) can be calculated. As earlier stated, two local hypotheses can not be present in the same global hypothesis if they share a measurement. The BGH should also include all the measurements

made from scan $k - N$ forward, where N is the size of the pruning window (see subsection 2.3.6. Under these conditions, there is a number of ways that the problem can be solved.

It is known that the problem of finding the Best Global Hypothesis can be reduced to the MDA (Multi-Dimensional Assignment) problem, which is a generalization of the assignment problem (a 2-dimensional problem). Despite this problem being NP-Hard when there are more than 2 dimensions, it has good approximations when solved by using Lagrangian Relaxation (Poore and Robertson III, 1997) and approximate linear programming (Storms and Spieksma, 2003).

A more recent work (Kim et al., 2015) reduces the problem to a MWIS (Maximum Weighted Independent Set), which is shown to be equivalent to the MDA problem. In this reduction, a graph is created where each node corresponds to a track hypothesis and its weight is the track's score. An edge is created between all the pairs of tracks that cannot coexist in the same global hypothesis due to a shared measurement in one or more time steps. In this case, finding the MWIS is equivalent to finding the valid global hypothesis that has the maximum track score. In this thesis, it was chosen to solve the Minimum Weighted Independent Set problem to find the Best Global Hypothesis.

Although Kim's work uses optimized algorithms to solve the problem, a more naïve solution was implemented in thesis due to the limited amount of time to learn the more complex implementations. In this solution, all the independent sets are recursively enumerated and compared until the one with the lowest cost is found. Despite simple to implement, this alternative requires a lot of processing, as in a scenario with N local hypotheses, there are 2^N subsets them that are potential global hypotheses.

Once the Best Global Hypothesis is calculated, it can be used as an estimate of how the tracks of the observed targets are. It is also used in conjunction to N-scan pruning to resolve or discard tentative tracks.

2.3.5 Gating

In gating, measurement is associated with a target only if it lies within its validation region, or gate (Reid, 1979). This way, only the measurements that are the most likely to be a continuation of a track are associated to it. If $\hat{x}$ and $\hat{P}$ are, respectively the predicted mean and covariance of the target, a measurement z lies within the validation region if:

$$(z - H\hat{x})^T S^{-1} (z - H\hat{x}) \leq \eta^2 \tag{2.22}$$

, where S is the innovation covariance calculated at 2.4 and η is the number of standard deviations of size for the validation region.

Suppose that there are two existing tracks T_1 and T_2, whose predicted measurements $\hat{T}_1$ and $\hat{T}_2$ are depicted in Figure 2.2 where it is assumed that the measurements are made in a 2D space. In the figure, the sets $\{z_1, z_3\}$ and $\{z_1, z_2\}$ correspond to the measurements that are valid as the continuations of the tracks T_1 and T_2, respectively, as they are within their validation regions. This way, only the data associations between tracks and measurements within their validation regions are considered.

2.3.6 N-Scan Pruning

This technique is a way of "consolidating" some local hypotheses while discarding others after a number of scans. In N-Scan Pruning, after the Best Global Hypothesis is calculated, the local

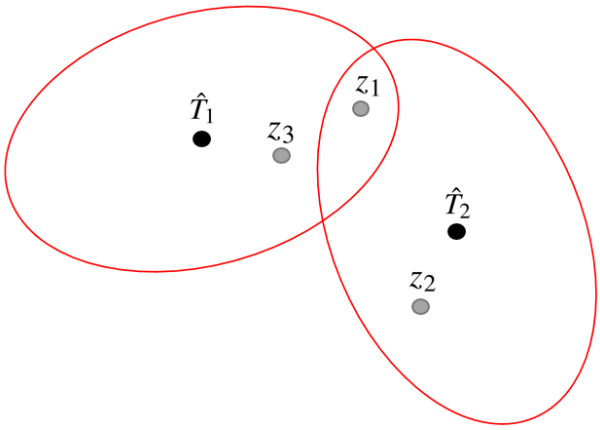


Figure 2.2: Simple gating example in a 2-D space (ignoring the dimensions that correspond to the velocity of the object). The two black dots correspond to mean of the predictions of the next measurements of two existing tracks. The red ellipses that surround the tracks represent the validation region, within which measurements are considered to be inside the gate. The three gray points are the measurements made at the current time step.

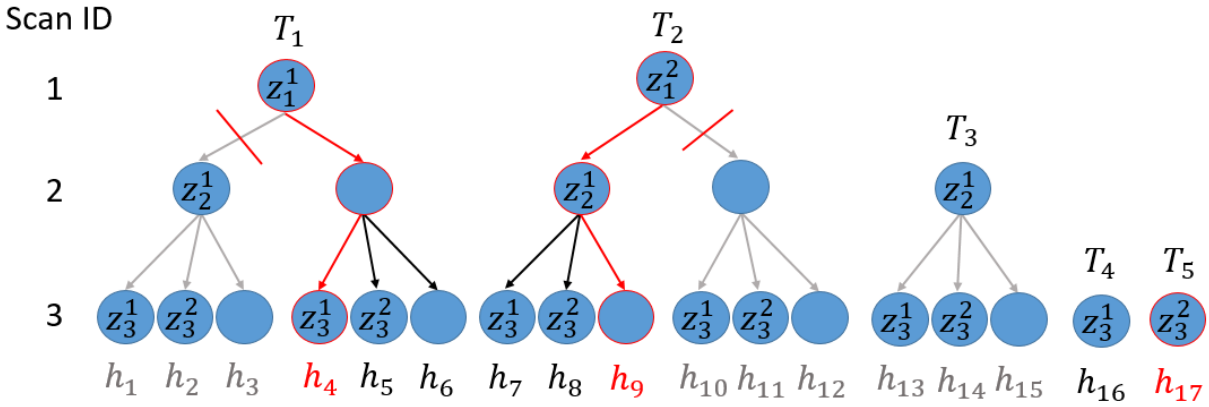


Figure 2.3: Example of N-scan pruning based on best global hypothesis. In this example, the size of the scanning window is 1. The BGH is composed of the hypotheses highlighted in red. All the hypotheses that diverge from the BGH in the previous scan are pruned and their identifications are represented in a dim tone of gray. Notice that the whole track T_3 is removed, as none of its hypotheses are considered.

hypotheses that diverge from it in the Nth previous scan are discarded. This also associates measurements to tracks in a definitive manner after the N-scan window.

Take Figure 2.3 as an example. In this case, the best global hypothesis is composed by the hypotheses $\{h_4, h_9, h_{17}\}$. Supposing $N = 1$, the algorithm resolves the track up to the 1st previous scan, which means that hypotheses $\{h_1, h_2, h_3, h_{10}, h_{11}, h_{12}, h_{13}, h_{14}, h_{15}\}$ are discarded.

N-scan pruning is an efficient way of slowing down the growing of number of hypotheses while still keeping many association hypotheses to the tracks.

3 STATE OF THE ART

The state-of-the-art research on Multiple Object Tracking focuses on techniques for object detection, data association and motion modeling. Beyond the themes covered by this thesis, computer vision is frequently used to acquire the measurements that serve as input for the tracking algorithms.

Currently, one of the best-performing algorithms for MOT is ByteTrack (Zhang et al., 2022). This algorithm, as the big majority of the others, does the data association between the bounding boxes of the detected objects instead of mere points. This algorithm is generalized for 2D and 3D object tracking, but the explanation will focus on he 2D case.

The algorithm solves the problem in 3 modules: detection, motion prediction and data association. The detection of the objects is made by the YOLOX detector (Ge, 2021). Each bounding box generated by the detector has a score, which is usually higher for clearly detected objects and lower for occluded objects or false alarms. The high-scored boxes give birth to the first tracks.

For motion prediction, the Kalman Filter is used. The state in the 2-D case corresponds to an eight-dimensional vector with the coordinates of the opposing corners of the bounding box and their velocities.

In the next time frames, the algorithm first tries to associate the newly-detected high-scored boxes with the existing tracks. Then, the low-scored boxes are associated with the remaining tracks. This part makes this algorithm different than the other ones, as the low-scored boxes are usually discarded, rather than associated. For the association, a similarity metric is used between the tracks and the detections. The metric can be based on appearance or on how much the boxes intersect. As opposed to MHT, the ByteTrack algorithm does only 2-dimensional data association, which means that it only involves two consecutive time steps. The DAP in this case is solved by the Hungarian algorithm (Kuhn, 1955).

According to Azevedo (2022), ByteTrack achieved state-of-the-art results in many different benchmarks, outperforming all the other trackers compared in the work both in how well and how fast the objects are tracked. The work of Abouelyazid (2023) compares the ByteTrack with the SORT and DeepSORT algorithms and also observes its faster speed and better tracking performance over the other ones.

Another very well performing MOT algorithm is StrongSORT++ (Du et al., 2023). It derives from the Simple Online and Real-time Tracking (SORT) algorithm, which praises simplicity, speed and effectivity. The SORT algorithm uses the Kalman Filter, associates data with the Hungarian algorithm using the overlap between predicted and detected bounding boxes. The high amount of identity switches (mistaken object associations) motivated the creation of DeepSORT, which integrates appearance information to SORT and significantly reduces the problem.

By changing DeepSORT's detector to YOLOX, using the novel techniques of ByteTrack and deep person re-identification (ReID), a technique for identifying the same person over multiple time frames, StrongSORT was created. This novel algorithm already achieves SOTA tracking performance. Nevertheless, the authors go even further and propose the addition of two plug-and-play algorithms to refine the results. The first one is the appearance-free link model (AFLink) to address the problem of an object being present in more than one track, and the second one is the Gaussian-Smoothed Interpolation (GSI), which uses the Gaussian process regression algorithm to compensate for missed detections.

These two improvements originated StrongSORT++, an algorithm that achieved SOTA status in many different benchmarks. While ByteTrack still achieves slightly better overall accuracy than StrongSORT++, the second outperforms the first in data association correctness.

4 IMPLEMENTATION

This chapter explains the implementation of the Track-Oriented Multiple Hypothesis Tracking algorithm done in this work. The structure of the algorithm, as well as some pseudo-codes will be used to clarify the overall structure used to apply the proposed solution.

4.1 IMPORTANT DATA STRUCTURES

In this implementation of the Track-Oriented Multiple Hypothesis Tracking algorithm, two data structures were created and are going to be mentioned in the algorithm description: the track tree and the compatibility graph.

4.1.1 Local Hypothesis Tree

The local hypothesis tree is the tree structure that stores local hypotheses. Each node of the tree is composed of:

- *id*: Unique identification number for the node in its time frame.
- *parent*: Parent node (or empty if it is a root node).
- *children*: A list of all the node's children.
- *cost*: The cost (NLLR) of the track.
- z: The measurement that gave origin to the node (or no measurement for missed detections).
- $\hat{x}_{k|k}$: The estimated state of the target calculated in the Kalman update.
- $P_{k|k}$: The covariance matrix for state of the target calculated in the Kalman update.
- $\hat{x}_{k+1|k}$: The predicted state of the target calculated in the Kalman prediction.
- $P_{k+1|k}$: The covariance matrix for the predicted state of the target calculated in the Kalman prediction.

This structure allows access from one of the nodes in a tree to all of its other nodes. Each leaf of a track tree is a track hypothesis in the current time step. The whole trajectory of a track hypothesis can be generated by getting the estimated states of the target from the root until the leaf.

4.1.2 Compatibility Graph

The idea of a compatibility graph came from Kim et al. (2015), where the Best Global Hypothesis is found by solving the Minimum Weighted Independent Set problem.

In this thesis' implementation, the compatibility graph is a boolean adjacency matrix and each of its nodes corresponds to the identification number of a track hypothesis. A cell of the matrix is true if the tracks corresponding to its line and column are connected (incompatible) and false if not (compatible).

An auxiliary vector is used to access the local hypotheses and their scores.

4.2 PARAMETER SETUP AND INPUT

In this implementation, the setup of the algorithm includes the following parameters that can be tuned to model the tracking scenario:

- N: Size of N-scan pruning window.
- P_D: Probability of detection of the targets. Assumed to be the same for every target.
- λ_ν: Scalar number that corresponds to the spacial density of new targets in a Poisson distribution.
- λ_ϕ: Scalar number that corresponds to the spacial density of false alarms in a Poisson distribution.
- T: Time frame between scans.
- $\dot{x}_{\max}$: Maximum target speed. Used to initialize prior probability distribution of the target state.
- $n\_dimensions$: Number of spacial dimensions that the target moves. If, for example, the target is on the ground, two dimensions are likely enough. Otherwise, if airplanes are being tracked in the sky, three dimensions can be necessary.
- η: Gate size. Corresponds to the number of standard deviations of tolerance to consider a new measurement as the continuation of a track.

The following Kalman Filter matrices are defined in a text file:

- H: measurement model matrix,
- R: measurement noise covariance matrix,
- F: motion model matrix,
- Q: process noise covariance matrix.

The algorithm assumes a constant time interval between scans, and this time interval is expressed in the motion model matrix, as seen in Equation 2.2.3.

Each of the matrices in the text file is first described by its dimensions, followed by all of its elements, all separated by white spaces. For example, a 2×4 would be represented as:

```
4 2
1 0 0 0
0 0 1 0
```

. Following the Kalman Filter matrices, the text file contains the input for the algorithm, i.e. the measurements made at each time step. This part begins with the total number of time frames that will be processed. After that, each time frame starts with its number of measurements, followed by all the measurements. Each measurement is represented by a sequence of numbers separated by white spaces. The amount of numbers corresponds to the number of dimensions in the measurement vector. The overall structure of this part is:

```
[Number of time frames]

[Number of measurements at time 1]
[measurement 1]
[measurement 2]
...

[Number of measurements at time 2]
[measurement 1]
[measurement 2]
...

...
```

An example of a measurement input is:

```
3

2
0.5 -0.5
10 3.2

0

3
-1 2
2 -4
-3 -6
```

4.3 MAIN CYCLE

In this implementation of the algorithm, there is a main cycle where the scans of the sensor are made. The sensor is expected to produce only one measurement per tracked object and each measurement is assumed to correspond to only one object (false alarms and missed detections are expected). The types of objects being tracked are not relevant to the algorithm, as they are represented by vectors of numbers. In the case of this implementation, the sensor data used as input is read from a text file.

The cycle is a loop that receives as input the current set of measurements and performs the following actions:

1. Kalman Prediction: Predicts the mean and covariance of the states of the tracks considering the difference in time T from the last scan.

2. Track hypothesis update: Creates three types of new tracks for the current scan: Track continuations (with use of gating), missed detections and new tracks. It also calculates all the new track scores.

3. Form best global hypothesis: Calculates the best hypothesis (combination of compatible tracks with highest score)

4. Track management: Perform N-scan pruning to exclude tracks not closely related to best hypothesis.

5. Kalman Update: Uses the Kalman filter to set the state and covariance of the new tracks based on the predictions made by their parents.

The **Kalman Prediction** loops into all the existing local hypotheses and predicts their next states based on their current estimations. For each of the hypotheses, Algorithm 1 is run and returns the predicted state and covariance.

Algorithm 1 Kalman Prediction

Input: $\hat{x}_{k|k}$: Current state estimation, $P_{k|k}$: current covariance estimation, F: motion model matrix, Q: process noise covariance.
Output: $x_{k+1|k}$: New state estimate, $P_{k+1|k}$: new covariance estimate for the state
calculate $x_{k+1|k}$ according to 2.8
calculate $P_{k+1|k}$ according to 2.9
return $x_{k+1|k}$ and $P_{k+1|k}$

The predictions of the track hypotheses, in conjunction with the current list of sensor detections are then used to **update the track hypotheses** by expanding the existing track trees and also creating new ones. In this step, each combination of a new detection and a track hypothesis is evaluated through gating (Equation 2.22) and only associations where the measurement lies within the gate are consolidated.

Consolidating an association between a track hypothesis t and a detection z means creating a new branch in the local hypothesis tree from the track hypothesis t to measurement z.

In this phase, the graph of track compatibility is also updated. Two tracks are incompatible if they share at least one measurement made any time step. The set of tracks incompatible with a given track t at a time step k is entirely composed of two possibly intersecting subsets:

1. The set of local hypotheses that were already incompatible with t at time step $k-1$,

2. The set of local hypotheses share a measurement with t at time step k.

The first subset is composed of all the tracks whose parent track was not compatible with the parent of t in the previous compatibility graph and also the tracks that share their parent with t. This subset is computed by a separate function illustrated by Algorithm 2

The second subset is more straightforward and can be easily obtained by creating a list of tracks who share the last measurement with t. As it is shown in Algorithm 3, instead of generating the second subset for each of the created track hypotheses, a list of all the hypotheses that share a determined measurement z is first created and then a clique is made in the compatibility graph to connect all the hypotheses in the list.

The chosen implementation for the track hypothesis update follows Algorithm 3.

After the track trees are updated, the **Best Global Hypothesis is calculated**. As mentioned in subsection 2.3.4, there are some optimized ways to find the BGH, or one of the best. Given that the learning curve of these algorithms and the limited amount of time to implement them, it was chosen use the naïve solution instead, which consists of finding all the valid global hypotheses and calculating the one with the least cost, as shown in the recursive Algorithm 4.

In the first call to Algorithm 4, *CGH* is an empty global hypothesis, while T is equal to the set of all the current track hypotheses. At each recursive call, one element is taken out of T and two recursive calls - one where t is inserted in *CGH* and one where it is not are made. When

Algorithm 2 Get Previously Incompatible Tracks

Input: t: Current track hypothesis, *CPrev*: Previous compatibility graph
Output: *incompatible*: List of track hypotheses incompatible with t

```
incompatible ← []
parent ← parent track hypothesis of t
for neighbor in neighborhood of parent in CPrev do
  for child in neighbor.children do
    push child to incompatible
  end for
end for
for sibling in children of parent do
  if sibling is not t then
    push child to incompatible
  end if
end for
```

Algorithm 3 Update track hypotheses

Input: Z: list of new detections, T: Current list of tracks, C: Current graph of incompatible tracks
Output: *TNew*: Updated list of tracks, *CNew*: updated graph of incompatible tracks

```
TNew ← []
CNew ← empty graph
for measurement z of Z do
  for track t of T do
    if z is within the gate of t (Eq. 2.22) then
      Create new track branch tc from track t with measurement z
      Calculate track score of tc
      Insert tc in TNew
      Insert tc in CNew and add edges to incompatible tracks (Alg. 2)
    end if
  end for
  Create new track tree root tn with measurement z
  Calculate track score of tn
  Insert tn in TNew
  Insert tn in CNew
  Make clique in CNew connecting all tracks generated in this iteration
end for
for track t of T do
  Create new track branch tm from track t without a measurement
  Calculate track score of tm
  Insert tm in TNew
  Insert tm in CNew and add edges to incompatible tracks
end for
```

Algorithm 4 Calculate best global hypothesis

Input: *CGH*: Current global hypothesis being built, *T*: Set of track hypotheses that can be inserted in *CGH*
Output: *BGH*: Best global hypothesis
  if *T* is not empty **then**
    $t \leftarrow$ first element of *T*
    incompatible_t $\leftarrow$ tracks in *T* incompatible with *t*
    Remove *t* from *T*
    GH1 $\leftarrow$ return from Algorithm 4 using *CGH* and *T*
    Remove *incompatible_t* from *T*
    Insert *t* in *CGH*
    GH2 $\leftarrow$ return from Algorithm 4 using *CGH* and *T*
    Return less costly global hypothesis between *GH1* and *GH2*
  else
    Return *CGH*
  end if

t is in *CGH*, all the track hypotheses that are not compatible with it are removed from *T*, since inserting them would lead to a non-valid global hypothesis.

After calculating the best global hypothesis without and with *t*, both are compared and the one with the least cost is returned. An empty global hypothesis has a positive infinite cost.

When *T* is empty, no more track hypotheses can be added to the global hypothesis and, therefore the resulting global hypothesis is returned. Once the BGH is known, the local hypotheses that are too divergent are pruned in the N-scan Pruning phase.

The **N-Scan Pruning** algorithm iterates through all the tracks in the BGH and performs pruning in the Nth ancestor scan for each track, if it exists. Afterwards, track trees that do not contain any track in the BGH are entirely deleted. It is divided in three parts: Track hypothesis pruning (Algorithm 5, Nth ancestor pruning (Algorithm 6) and N-scan pruning (Algorithm 7).

N-scan pruning (Algorithm 7) is itself the implementation of N-Scan Pruning, calling Algorithm 6 for each track hypothesis in the BGH to remove all its diverging relatives in the Nth previous scan and afterwards calling Algorithm 5 to prune all the roots of the tracks not present in the BGH.

Nth ancestor pruning (Algorithm 6) receives a track hypothesis *t*, goes up *N* generations in the track tree and calls Algorithm 5 to prune each of its children, except for the one that leads to track *t*.

Track hypothesis pruning (Algorithm 5) deletes a track by recursively removing all of its children and then deletes itself. It also reflects the deletions in the set of tracks and the compatibility graph.

The **Kalman Update** is straightforward, as it simply applies the Kalman Update equations, as seen in Algorithm 8.

4.4 OUTPUT

After executing a scan, the algorithm outputs the current Best Global Hypothesis. It consists of one line for each of the track hypotheses. Each track hypothesis is represented by the sequence of the nodes in the tree, from root to leaf, which corresponds to the object trajectory. Each of the nodes has the structure

Algorithm 5 Prune track hypothesis

Input: t: track hypothesis to be pruned
  if t does not have children **then**
    remove t from set of tracks
    remove t from compatibility graph
  else
    for *child* in children of t **do**
      prune *child* with this algorithm
      delete *child*
    end for
  end if

Algorithm 6 Prune Nth Scan

Input: t: track from most likely hypothesis, N: Number of nodes left to go up, p: protected child branch
  if $N < 0$ **then**
    prune all children of t different than p by Algorithm 5
  else if t has a parent **then**
    call this algorithm with parameters T = parent of t, $N = N - 1$, $p = t$
  end if

Algorithm 7 N-Scan Pruning

Input: H: Current most likely hypothesis, *Roots*: Roots of the current tracks, S: Size of scan pruning window.
Output: *newRoots*: New root set
  newRoots $\leftarrow$ []
  for t in H **do**
    Insert root of t in *newRoots*
    Run Algorithm 6 with parameters $t = t$, $N = S$, $p = t$.
  end for
  for *root* in *Roots* and not in *newRoots* **do**
    prune *root*
    delete *root*
  end for
  Roots $\leftarrow$ *newRoots*

Algorithm 8 Kalman Update

Input: x: Current state prediction, P: current covariance prediction, z: newest detection, H: measurement model matrix, R: Measurement noise covariance.
Output: x: New state estimate, P: new covariance estimate for the state
  calculate ν_{k+1} according to 2.3
  calculate S_{k+1} according to 2.4
  calculate W_{k+1} according to 2.5
  calculate $x_{k+1|k+1}$ according to 2.6
  calculate $P_{k+1|k+1}$ according to 2.7
  return $x_{k+1|k+1}$ and $P_{k+1|k+1}$

```
[Scan ID, Measurement ID, Cost of the track (NLLR)].
```

Suppose the scenario where three consecutive scans give the following measurements:

$$Z_1 = \{z_1^1, z_1^2\}$$
$$Z_2 = \{z_2^1\}$$
$$Z_3 = \{z_3^1, z_3^2, , z_3^2\}.$$

Then, a possible output for the algorithm would be:

```
T1: [1, 1, 0.2], [2, 1, 0.35], [3, 2, 0.25]
T2: [1, 2, 0.2], [2, 0, 0.7], [3, 2, 0.85]
T3: [3, 1, 0.2]
```

Note that, in track T2, scan 2, the ID of the measurement is 0. This means that this hypothesis considers that target 2 was not detected at time step 2.

The outputted data can easily be changed to also display other information from the hypothesis, such as estimated state.

5 METHODS

In this chapter, it will be explained how the algorithm was tested. All the experiments were run with an Intel I7 6700 processor with a 3.4 GHz base frequency, running the Windows 10 operating system.

The tests were made assuming that the objects move in two dimensions. The state of an object consists of a 4-dimensional vector with two being the coordinates of its position and the other two corresponding to the velocities on each of the axes:

$$\mathrm{x} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}$$

The measurements are two-dimensional vectors, each dimension corresponding to the object coordinates in each of the axes.

$$\mathbf{z} = \begin{bmatrix} x \\ y \end{bmatrix}$$

.

The measurement model matrix is, therefore,

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

.

All of the tests consist of static objects or objects with constant velocity in both dimensions. The time frame between scans is $T = 2$ seconds. Therefore, for all of the tests, the constant motion model matrix was used:

$$F = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The process noise covariance matrix was

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

while the measurement noise covariance matrix was

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

The probability of detection used in the tests was $P_D = 0.90$, the target appearance and false alarm densities were $\lambda_\nu = 0.2$, $\lambda_\phi = 0.002$, the maximum target speed was $\dot{x}_{\max} = 2$ and the N-scan pruning window was $N = 2$.

Given that the implementation of the function that generates the Best Global Hypothesis is of exponential nature with respect to the number of local hypotheses, the growth of this number can result in a significant increase in the processing time. Therefore, all of the tests consisted of 4 or 5 time frames. The tests were divided into two categories: **crafted tests** and **random batched tests**.

The **crafted tests** were manually created to test specific scenarios. These scenarios were built to test if the algorithm can solve simple problems and progressively increase their complexity, adding missed detections and false alarms to them. The tests consisted of the following categories:

Single static target: In this test, the target does not move through space and always generates a measurement at the same spot for three time steps. In this case, the detection input consisted of three ordered pairs $(1, 1)$.

Single moving target: In this test, the target moves in a linear motion through the space of detection during 4 iterations with a small noise added to the measurements. In the following example, the target's velocity in the two axes are respectively 2 and 1.

```
1
0 0

1
2.01 1.01

1
3.98 2.01

1
6 2.99
```

Single moving target with missed detection: In this test, the target is detected in all time steps but 3. The input is analogous to the single moving target without the detection at time step 3.

```
1
0 0

1
2.01 1.01

0

1
6 2.99
```

Single moving target with false alarms: In this test, besides the moving target, false alarms were introduced at time steps 2 and 4.

```
1
0 0

2
```

```
2.01 1.01
1 1

1
3.98 2.01

2
6 2.99
-2 2
```

Single moving target with missed detection and false alarms: In this test, there are both false alarms at time steps 2 and 4 and a missed detection at time step 3.

```
1
0 0

2
2.01 1.01
1 1

0

2
6 2.99
-2 2
```

Two crossing targets: In this test, two targets move towards each other. Between the third and fourth frame, the targets cross. In this test case, the tracker is expected to have estimated the velocity of the target by the third frame in a way that it can correctly associate the detections in the fourth frame to their tracks.

```
2
-4 4
-4 -4

2
-2 2.5
-2 -2

2
0 1
0 0

2
2 -0.5
2 2
```

The **random tests** were created to test different scenarios in batches of random positions and speeds. The tested scenarios were the following:

- Three targets, four time steps.

- Three targets, four time steps, missed detection for one of the targets at time step 3.
- Two targets, five time steps.
- Two targets, four time steps, a false alarm at time 2 and 4.
- Two targets, four time steps, missed detections at time 3 and one false alarm at times 2 and 4.
- Three targets, five time steps.

Each of the scenarios was generated 10 times. In each instance of a scenario, the positions and the velocities of the targets were randomly generated.

The initial position of a target consists of two random numbers between -5 and 5, each corresponding to the object coordinate in each axis. The velocity of the target is composed of two random numbers between $-\sqrt{2}$ and $\sqrt{2}$, one for each axis. This way, the maximum possible velocity of each target can be 2. Once the initial position and the velocity of the targets is chosen, the measurements can be easily generated at each time step by the constant velocity formula. A small noise was also added to each of the measurements to simulate the noise that occurs in the real world. For practical purposes, the noise follows a uniform distribution between -0.2 and 0.2 for each of the axes.

When a missed detection is expected for a target at a given time step, the corresponding measurement is not included in the input. The false alarms are uniformly distributed, and each of their coordinates can assume values between -10 and 10.

6 RESULTS AND EVALUATION

6.1 DATA ASSOCIATION

With respect to the Data Association Problem, the algorithm performed as expected in all of the test cases for the trajectories of the real objects. This means that the algorithm correctly detected the trajectories of the real objects.

This can be seen in the results of the first crafted tests: single static target and single moving target, whose inputs are shown in chapter 5. Their outputs were, respectively,

```
T1: [1, 1, -4.61], [2, 1, -2.61], [3, 1, -1.50]
```

and

```
T1: [1, 1, -4.61], [2, 1, -2.13], [3, 1, -0.96], [4, 1, -0.08]
```

. Although these tests and results are trivial, they show that the algorithm is capable of associating all the measurements to only one track, with neither false alarms nor missed detections. The more complex crafted scenario with two crossing targets shown in Chapter 5 was also correctly tracked and yielded the following output.

```
T1: [1, 1, -4.61], [2, 1, -1.91], [3, 1, -0.64], [4, 1, 0.28]
T2: [1, 2, -4.61], [2, 2, -1.74], [3, 2, -0.45], [4, 2, 0.49]
```

In the tests that included **missed detections**, the data association was correctly made. The output for the single moving target with missed detection was the following:

```
T1: [1, 1, -4.61], [2, 1, -2.02], [3, 0, 0.28], [4, 1, 2.66]
```

As expected, the test case correctly predicted only one track with no missed detections. This is expected not only because of the track prediction, but also because two consecutive missed detections are not allowed. Notice that the cost of the track is higher than the previous example. This happened because of the high probability of detection (low probability of missed detection), in conjunction with the loss of information in the third time frame. Nevertheless, as will be shown in the subsection that explains target trajectories, the algorithm was able to roughly predict the direction that the target was heading. In a more complex example, there are three targets and one of them has a missed detection at time frame 3:

```
3
-0,24-1,78
-3,03-4,69
-4,73-3,30

3
0,52-0,99
-1,90-3,40
-4,52-2,31
```

```
2
1,29-0,19
-0,78-2,11

3
2,050,60
0,35-0,82
-4,11-0,33
```

The output for this scenario was the following:

```
T1: [1, 1, -4.61], [2, 1, -2.39], [3, 1, -1.20], [4, 1, -0.29]
T2: [1, 2, -4.61], [2, 2, -2.23], [3, 2, -1.01], [4, 2, -0.10]
T3: [1, 3, -4.61], [2, 3, -2.13], [3, 0, -0.11], [4, 3, 2.20]
```

The targets in this case were also correctly tracked, including the one with a missed detection.

In the test cases with only **false alarms**, the expected result would be for them to be discarded and not considered as parts of real tracks. Nevertheless, they were usually considered as separate targets and were not entirely excluded, as will be shown in the following example.

In the test case with a single moving target and false alarms, false alarms were introduced at time steps 2 and 4 besides the measurements generated by the moving target, as shown in Chapter 5. The tracks generated by this test case were the following:

```
T1: [1, 1, -4.61], [2, 1, -2.13], [3, 1, -0.96], [4, 1, -0.08]
T2: [2, 2, -4.61], [3, 0, -1.61], [4, 2, 1.51]
```

Track `T1` remained exactly the same as the previous test, but another track `T2` was created with the false alarms. Another example where the same problem is present is the input with two randomly generated targets with false alarms at time steps 2 and 4. At each time step, the first measurement is generated by the first object and the second measurement is generated by the second object. At time steps 2 and 4, the third measurements (z_2^3 and z_4^3) are false alarms.

```
4

2
-2.45 0.30
-4.48 -3.32

3
-2.80 -0.23
-5.37 -2.89
9.01 4.18

2
-3.16 -0.75
-6.27 -2.47

3
```

```
-3.51 -1.28
-7.16 -2.04
3.25 -8.87
```

The output for the previous input consists of the following detected tracks.

```
T1: [1, 1, -4.61], [2, 1, -2.58], [3, 1, -1.45], [4, 1, -0.6]
T2: [1, 2, -4.61], [2, 2, -2.52], [3, 2, -1.39], [4, 2, -0.53]
T3: [2, 3, -4.61], [3, 0, -1.61], [4, 3, 28.75]
```

Again, track `T1` correctly consists of the first measurement at each of the time steps and track `T2` does the same for the second measurement. Track `T3`, on the other hand, does not correspond to measurements of real objects, but is only composed of false alarms or missed detections. Note that the cost of `T3` increases from -1.61 to 28.75 between time steps 3 and 4. This is expected, given the large distance between the measurements that compose the track.

A plausible reason for the false-alarm tracks in both the previous examples is that the Best Global Hypothesis generation includes all the measurements made in the N-scan window (including false alarms) and no pruning technique was implemented for tracks with excessively high costs, the false alarms were not discarded and were considered as a new track. The same pattern repeats in other scenarios that contain false alarms.

When **false alarms** and **missed detections** are present at the same time, the tracks are still correctly identified and a new track with the false alarms is created, instead of them being discarded. The **single moving target with missed detections and false alarms** scenario shown in Chapter 5 yielded the following output.

```
T1: [1, 1, -4.61], [2, 1, -2.02], [3, 0, 0.28], [4, 1, 2.66]
T2: [2, 2, -4.61], [3, 0, -2.30], [4, 2, 1.41]
```

With two real targets, similar results were obtained, as shown in the following input example.

```
2
0,81 1,71
3,53 2,74

3
1,62 1,99
3,04 2,97
-1,25 5,61

0

3
3,23 2,55
2,06 3,45
0,66 -0,06
```

This input yielded the following output.

```
T1: [1, 1, -4.61], [2, 1, -2.43], [3, 0, -0.14], [4, 1, 2.17]
T2: [1, 2, -4.61], [2, 2, -2.48], [3, 0, -0.18], [4, 2, 2.12]
T3: [2, 3, -4.61], [3, 0, -2.30], [4, 3, 2.17]
```

The output is a mix of the results obtained from the separate scenarios with only false alarms and only missed detections. The real targets were correctly tracked and a third with the false alarms was created.

6.2 ESTIMATED TARGET TRAJECTORIES

This subsection has the purpose to illustrate the detected trajectories of the targets in the different scenarios tested in this thesis. The estimated trajectory of a target is the sequence of updated states that result from the calculations made by the Kalman Filter. In scenarios where there were no missed detections, the estimated target trajectories were very close to the measurements given as input to the algorithm. Take as example one of the targets in the scenario where two targets have crossing trajectories. For simplicity, the actual positions of the target correspond exactly to the measurements given as input to the algorithm. As the time frame between to scans is 2 seconds, the real velocity of the target corresponds to half the difference of two consecutive measurements.

Time step	Real position	Estimated position	Real velocity	Estimated Velocity
1	(-4.00,4.00)	(-4.00,4.00)	(1.00,-0.75)	(0.00,0.00)
2	(-2.00,2.50)	(-2.04,2.53)	(1.00,-0.75)	(0.77,-0.58)
3	(0.00,1.00)	(-0.02,1.02)	(1.00,-0.75)	(0.88,-0.66)
4	(2.00,-0.50)	(1.99,-0.49)	(1.00,-0.75)	(0.92,-0.69)

Table 6.1: Estimated target states of target 1 in the scenario of targets with crossing trajectories.

Table 6.1 compares the real and estimated target states (position and velocity) at all of the time steps. Notice that all the estimated positions of the target are close to its real positions. On the other hand, its estimated velocity gets closer to the real velocity as more scans are made. In this case, the estimation of the velocity has very small influence in the estimation of the object trajectory, since it is an ideal case with no missed detections. As it will be shown in the following example, the estimated target velocity is of crucial importance for the cases where missed detections are present.

In the last depicted scenario of the previous subsection, the first target has an initial position of $(0.81, 1.71)$ and a velocity of $(0.40, 0.14)$ and is not detected at time step 3. The following table compares the real states of the target and its estimations.

Time step	Real position	Estimated position	Real velocity	Estimated Velocity
1	(0.81,1.71)	(0.81,1.71)	(0.40, 0.14)	(0.00,0.00)
2	(1.61,1.99)	(1.59,1.98)	(0.40, 0.14)	(0.31,0.11)
3	(2.41,2.27)	(2.21,2.20)	(0.40, 0.14)	(0.31,0.11)
4	(3.21,2.55)	(3.20,2.55)	(0.40, 0.14)	(0.37,0.13)

Table 6.2: Estimated target states of target 1 in randomly-generated scenario with two targets with false alarms and missed detection.

Notice that Table 6.2 shows that, despite the target not being detected at time step 3, its estimated position was not very far from the real one. The estimated velocity played a crucial role in this estimation, since it indicates the direction and speed the target most probably moved.

6.3 PROCESSING TIME

A noticeable issue that arose during the testing phase was processing time. The random scenarios with three targets were the slowest to be processed. Each instance of the scenarios where 3 targets were tracked for 4 time steps took approximately 5 minutes to be processed. Meanwhile, the scenario with 3 targets and 5 time steps could not be processed in less than 1 hour and were therefore discarded.

Despite the measures used to mitigate the growth on the number of local hypotheses, i.e. Gating and N-scan Pruning, this number still grows with the number of time steps and measurements. As explained in Chapter 5, the processing time of the algorithm grows exponentially with the number of local hypotheses at a given time step. There are two possible solutions for this problem. The first one would be to implement a mechanism for limiting the number of local hypotheses so that the time to calculate the Best Global Hypothesis keeps under a given threshold. The second one would be to implement one of the more efficient solutions to find the Best Global Hypothesis mentioned in Subection 2.3.4.

6.4 CONCLUSION

In this section, it was shown that despite some failures, the implemented algorithm works on the most basic examples. The data association was correctly made for real targets in the presence of missed detections and/or false alarms.

One of the identified problems was that, due to the BGH always including all the measurements in the N-scan frame, false alarms were considered as a separate track instead of being discarded. A measure to mitigate this problem would be to create a threshold for the cost function of the track. Tracks with an excessively high cost are very unlikely to be real and, hence, should not be considered for a global hypothesis.

The other big obstacle was the processing time of the algorithm. The implemented function to generate the BGH has an exponential nature and, therefore, takes a longer time to be processed as the number of track hypotheses grows. Therefore, the current slow and naïve solution should be replaced by one of the faster and well-researched functions, e.g. the ones described in (Poore and Robertson III, 1997) and (Storms and Spieksma, 2003).

7 CONCLUSION AND OUTLOOK

The growing need for real-time tracking in dynamic systems, where accurate trajectory estimation of multiple objects is crucial, provided the primary motivation to develop this thesis. Multiple Object Tracking is a complex problem that has received increasing attention over the past few years with the improvement of computer vision techniques and Machine-Learning detectors.

The study, development, and evaluation of a functional and generic tracker that uses the TO-MHT algorithm to solve the MOT problem resulted in a foundational solution, which lays the groundwork for future improvements and optimizations. Techniques such as gating and N-scan pruning were employed to reduce the number of track hypotheses, and the data association process solved through the Minimum Weight Independent Subgraph problem. However, the naïve solution to this problem, using subset enumeration, became a bottleneck and affected the performance of the algorithm, especially as the number of detections and scans increased.

Despite these limitations, the tracker was successful in tracking objects even in the presence of false alarms and missed detections, showcasing its effectiveness in simple applications. Nevertheless, the inability to discard false alarms—treating them as separate tracks remains a challenge. Additionally, the processing time grew rapidly with the increasing number of detections and scans due to the time required for data association, which could limit scalability.

Overall, this thesis provides a foundation for the development of a TO-MHT tracker. Future work can focus on optimizing the data association process by implementing more sophisticated techniques to solve the Data Association Problem, such as the ones presented in Poore and Robertson III (1997), Storms and Spieksma (2003) and Kim et al. (2015). This way, the algorithm can track objects in a much faster way and can be tested with more complex datasets that have more targets to be tracked, more false alarms and that last for more scans. Another point for improvement is the handling of false alarms by discarding them instead of considering them as separate tracks. This could be done by adding more pruning techniques, such as defining a maximum threshold for track cost.

Beyond the scope of the subjects included in this thesis, the tracker could also include computer vision techniques to be able to receive a video of the moving objects as input and show identified bounding boxes around the tracked objects as output.

These improvements can contribute to the scalability and performance of the tracker, and include features that make it useful in more practical applications. This enables its use in larger, more complex and real-world tracking systems in dynamic environments.

REFERENCES

Abouelyazid, M. (2023). Comparative evaluation of sort, deepsort, and bytetrack for multiple object tracking in highway videos. *International Journal of Sustainable Infrastructure for Cities and Societies*, 8(11):42–52.

Azevedo, P. (2022). Object tracking state of the art 2022. Accessed: 2024-04-24.

Bar-Shalom, Y., Blackman, S. S., and Fitzgerald, R. J. (2007). Dimensionless score function for multiple hypothesis tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 43:392–400.

Bar-Shalom, Y., Li, X. R., and Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation*. John Wiley and Sons, Incorporated, 1 edition.

Bar-Shalom, Y., Willett, P. K., and Tian, X. (2011). *Tracking and data fusion*, volume 11. YBS publishing Storrs, CT, USA:.

Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., and Meng, H. (2023). Strongsort: Make deepsort great again. *IEEE Transactions on Multimedia*, 25:8725–8737.

Ge, Z. (2021). Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*.

Kim, C., Li, F., Ciptadi, A., and Rehg, J. M. (2015). Multiple hypothesis tracking revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 4696–4704.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

ngu Vo, B., Mallick, M., Bar-Shalom, Y., Coraluppi, S., Osborne, R., Mahler, R., and tuong Vo, B. (2015). Multitarget tracking. *Wiley encyclopedia of electrical and electronics engineering*.

Poore, A. B. and Robertson III, A. J. (1997). A new lagrangian relaxation based algorithm for a class of multidimensional assignment problems. *Computational Optimization and Applications*, 8:129–150.

Reid, D. (1979). An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854.

Storms, P. P. and Spieksma, F. C. R. (2003). An lp-based algorithm for the data association problem in multitarget tracking. *Computers & Operations Research*, 30(7):1067–1085.

Svensson, L. and Granström, K. (2019). Chalmersx: Multi-object tracking for automotive systems. Accessed: 2024-04-24.

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2022). Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision*, pages 1–21. Springer.