Technische Universität Berlin
Faculty IV Institute of Telecommunication Systems
Communication Systems Group Prof. Dr.-Ing. Thomas Sikora

Master Thesis

Development and Evaluation of a Real-time Capable Multiple Hypothesis Tracker

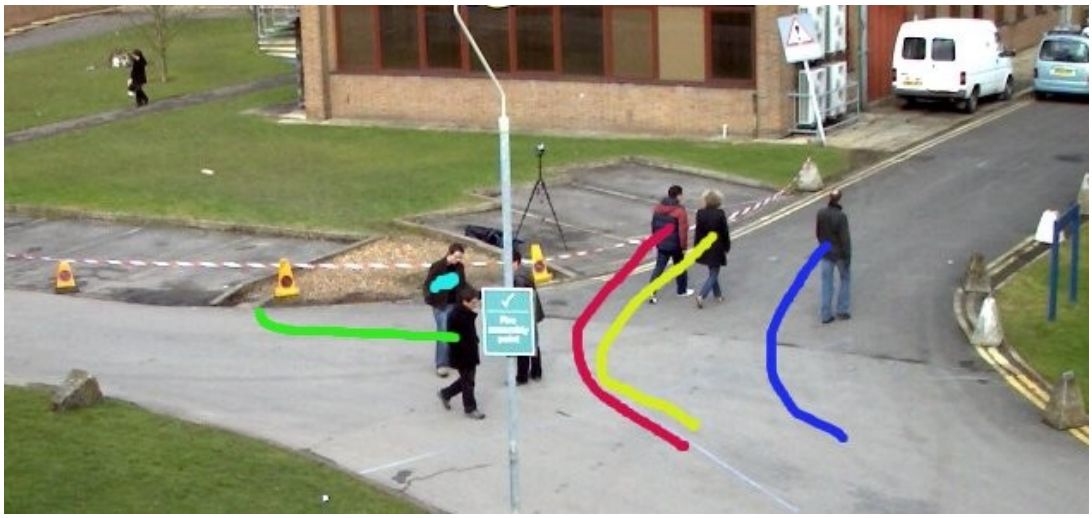
David Geier
david.geier@campus.tu-berlin.de
Matr.-Nr.: 336222

21. May 2012

Supervisors:
Dipl.-Ing. Michael Pätzold
Prof. Dr.-Ing. Thomas Sikora

Technische Universität Berlin
Fachgebiet Nachrichtenübertragung
Prof. Dr.-Ing. Sikora

- Master Thesis Propostition - „Development and Evaluation of a Real-time Capable Multiple Hypothesis Tracker“



One of the biggest challenges in the field of multi-target tracking is the association of a set of measurements originating from a sensor to a dynamic number of objects. Due to missing, noisy and false measurements caused by different kinds of interferences, an easy assignment is not possible in most cases. The combinatorial association problem can be solved by enumerating and evaluating all association possibilities in each time-step. On this principle the *Multiple Hypothesis Tracking* (MHT) algorithm is based. Due to its exponential complexity, the practical suitability of this method is highly dependent on the tracking scenario and the chosen parameters. In this thesis a real-time capable MHT system is to be developed and subsequently its suitability in the context of real-time video tracking is to be evaluated.

Scope of Work:

- Thorough literature research in the field of multi-target tracking
- Implementation of a basic MHT system
- Extension of the tracker with hypotheses clustering
- Optimization of the system through the implementation of a chosen algorithm in order to keep the size of the hypothesis set in check
- Choice of an appropriate measure for evaluating the system's performance with regard to the accuracy of the number and location of detected objects
- Evaluation of the system based on real-world simulation data from a video data-base with regard to the system's real-time capability

Michael Pätzold
Communication Systems Group
Technische Universität Berlin
Sekr. EN1
Einsteinufer 17
10587 Berlin
E-Mail: paetzold@nue.tu-berlin.de

Affirmation

Die selbständige und eigenständige Anfertigung versichere ich an Eides statt.

Ort, Datum	Unterschrift (David Geier)

Acknowledgements

First of all I would like to gratefully thank my supervisor Dipl.-Ing. Michael Pätzold who made this thesis possible. I especially thank him for all his fruitful ideas, advice and support but also for his flexibility and patience.

Additionally, I would like to thank Daniel Arp who let me use his evaluation framework and who gave me many tips concerning the evaluation.

Finally, I would like to say thank you to my parents who supported me my whole life and made my studies possible. As well I'm grateful to my girl-friend that supported me during many difficult days while working on this thesis.

Kurzfassung

Masterarbeit

Entwicklung und Evaluierung eines echtzeitfähigen
Multi-Hypothesen Trackers

David Geier

21. Mai 2012

Multi-Objekt-Verfolgung wird heutzutage in vielen wissenschaftlichen Disziplinen eingesetzt. Obwohl es seit Jahrzehnten wissenschaftlich untersucht wird, bleibt Multi-Objekt-Verfolgung wegen der hohen Komplexität ein schwieriges Problem. In der vorliegenden Masterarbeit wird die Echtzeitfähigkeit eines modernen Multi-Hypothesen Trackers für das Tracking von Personen in (Überwachungs-)Videos untersucht. Zu diesem Zweck wird zunächst die benötigte Theorie und anschließend die darauf aufbauende, im Rahmen dieser Arbeit entstandene, Implementierung vorgestellt. Besonderes Augenmerk wird bei der folgenden Evaluierung auf die Geschwindigkeit der Implementierung gelegt und es wird analysiert wo weiters Optimierungspotential besteht. Zusätzlich wird die Qualität des Trackings mit der eines Global Nearest Neighbor Trackers verglichen.

Abstract

Master Thesis

Development and Evaluation of a Real-time capable
Multi Hypothesis Tracker

David Geier

21. Mai 2012

Multi-object tracking is nowadays deployed in a vast number of scientific disciplines. Even though multi-object tracking has been researched for decades it still remains a challenging problem because of its computational complexity. The aim of this master thesis is to analyze the real-time capabilities of a modern multiple hypothesis tracker used for tracking persons in (surveillance) videos. For this purpose all required theory is discussed first and afterwards, the implementation created in the scope of this master thesis is presented. What follows is an evaluation which focuses on the speed of the implementation. Potential bottlenecks are analyzed and ideas for future optimizations are given. Supplementary, the tracking quality is analyzed and compared to the one of a global nearest neighbor tracker.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

β_{FT}	Density of false targets
β_{NT}	Density of new targets
$\mathrm{Cov}(\cdot)$	Covariance
$\dim(\cdot)$	Dimension of a vector
ϵ_{λ}	Hypothesis merging threshold for covariance matrix eigen values
ϵ_{sp}	Hypothesis merging threshold for state predictions
η	Size of gating volume
$\gamma(\cdot,\cdot)$	Lower incomplete Gamma function
$\hat{x}^{+}$	Kalman a posteriori state estimation
$\hat{x}^{-}$	Kalman a priori state estimation
λ	Clutter rate
λ_{max}	Maximum eigen-value for track deletion
$\log(\cdot)$	Natural logarithm
$\mathcal{O}$	Landau symbol for complexity
μ	Mean value of Gaussian distribution
Ω^{k}	Set of hypotheses
$\omega_{i}(k)$	New set of assignments
Ω_{i}^{k}	Hypothesis
$\mathrm{E}(\cdot)$	Expected value
$\overline{fp}$	Ratio of false positives

$\overline{mme}$	Ratio of mismatches
$\overline{m}$	Ratio of missed measurements
Σ	Covariance matrix of Gaussian distribution
MOTA$(\cdot,\cdot)$	CLEAR MOT accuracy metric for configuration errors
MOTP$(\cdot,\cdot)$	CLEAR MOT precision metric for localization errors
A	Set of all possible assignments
$a(\cdot)$	Measurement association probability
A_N	Restricted set of assignments of node N
C	Cost matrix
$c(\cdot)$	Cost of an assignment
C_N	Restricted cost matrix for node N
d_i^k	Localization error in CLEAR MOT metrics
D_M	Dimension of measurements
$d_M(\cdot,\cdot,\cdot)$	Mahalanobis distance
D_S	Dimension of system states
d_{λ_i}	Distance between covariance matrix eigen values
d_{sp}	Distance between track state predictions
E	Edges of a graph
e^+	Kalman a posteriori estimation error
e^-	Kalman a priori estimation error
E_N	Excluded cell in node N
$f_{\mathcal{B}}(\cdot)$	Bernoulli probability mass function
$f_{\mathcal{P}}(\cdot)$	Poisson probability density function
F_N	Fixed cell in node N
$F_{\chi^2}(\cdot)$	χ^2 cumulative distribution function
$f_{\mathcal{N}_p}(\cdot)$	p-dimensional Gaussian probability density function
G	Graph

G^k	Ground truth in time-step k
G_l	Equality graph of G with respect to labeling l
H	Kalman measurement matrix
H_k	Number of hypotheses
J	Measuring interval of sensor
K	Kalman gain or blend factor
L	Data association matrix for n-best MHT
$l(\cdot)$	Vertex labeling
L'	Modified data association matrix for direct n-best MHT
M	Matching in a graph
$m(\cdot)$	Index of parent hypothesis of given hypothesis
M_k	Number of measurements in scan
m_k	Number of misses
N	A node in Murty's algorithm
$N(\cdot)$	Neighbor set
N_i^k	Number of feasible posterior hypotheses
N_{DT}	Number of measurements assigned to previously existing tracks
N_{FT}	Number of measurements classified as false alarms
N_{NT}	Number of measurements that initiated new tracks
N_{TGT}	Number of tracks in a hypothesis
$P(\cdot)$	Probability
$P(\cdot,\cdot)$	Regularized Gamma function
$p(\cdot\|\cdot)$	System state's distribution
P_D	Detection probability
P_G	Gating probability
P_i^k	Hypothesis probability
Q	Kalman process noise covariance matrix

R	Kalman measurement noise covariance matrix
R	List of Murty result nodes
r	Kalman innovation vector
S	Kalman innovation covariance matrix
T	Sensor output to ground truth matching distance threshold
U	List of Murty nodes
u	Kalman control input
V	Scan volume covered by sensor
V	Vertices of a graph
v	Kalman measurement noise
w	Kalman process noise
$w(\cdot)$	Weight of an edge or matching
x	System state
$X \sim \chi^2(\cdot)$	A random variable χ^2-distributed
$X \sim \mathcal{B}(\cdot,\cdot)$	A random variable is Bernoulli distributed
$X \sim \mathcal{N}_p(\cdot,\cdot)$	A random variable is p-dimensional Gaussian distributed
$X \sim \mathcal{P}(\cdot)$	A random variable is Poisson distributed
X^k	Tracker output in time-step k
z	Measurement
$Z(k)$	Measurements contained in scan k
Z^k	Set of all scans up to time-step k
fp_k	Number of false positives
mme_k	Number of mismatches

1 Introduction

1.1 Motivation

Tracking is the fundamental element of all sorts of surveillance, guidance and obstacle avoidance systems. The term *tracking* refers to the problem of determining location, path and characteristics of one or multiple moving objects, so called *targets*, over time with the use of *measurements* from a *sensor*. A sensor, in that context, is any measuring device which can be used to collect information about targets in the *environment*, also referred to as *scan volume*. Typical examples of sensors are radars, cameras or infrared sensors [HL01].

Tracking is applied in a wide range of fields. Historically used in military and civil areas, tracking algorithms are nowadays even employed in molecular biology (see e. g. [SNM06]). In military use ballistic missile defense, air defense and battlefield surveillance can be mentioned. Tracking algorithms are employed nowadays for civil purpose in such contexts as air traffic control, ocean surveillance and surveillance of people in public places.
A practical example is the tracking of people in video sequences for automatic visual surveillance systems of human activity. Here it is attempted to determine the number of people in the environment under surveillance, their speed, positions and walking routes. All this information has to be obtained exclusively on the basis of the measurements output by the underlying camera sensor. Figure 1.1[1] shows the output of such a surveillance tracking system for people.

Unfortunately, there is a number of sources of *uncertainty* which makes the tracking

[1]The image was taken from `http://www.micc.unifi.it/wp-content/uploads/2010/07/scale-invariant-3d-multiperson-tracking.jpg`.

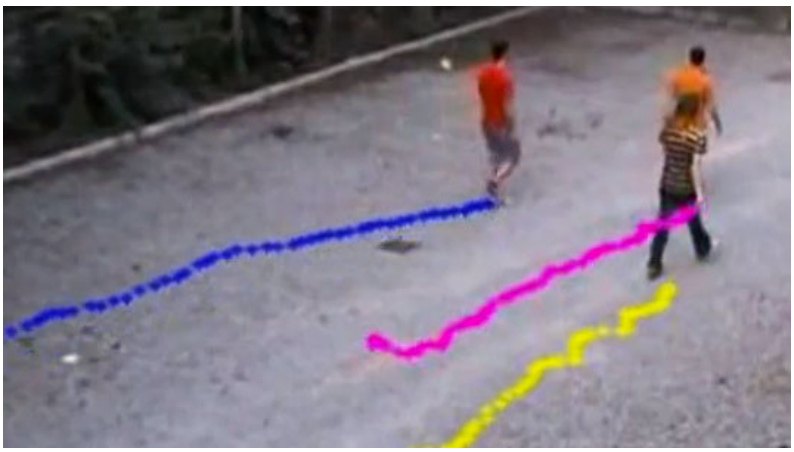


Figure 1.1: An example for the output of a visual surveillance tracking system for people.

task particularly challenging and difficult. Occurring problems are various but a few examples should be given. The motion of a target can be arbitrary and thus any motion model is likely to fail. The number of targets in the scan volume can change randomly between subsequent time-steps. It can happen that a target stays undetected for some time-steps because multiple targets move in close proximity and the resolution of the used sensor is limited. Another conceivable scenario is that a target is not detected by the sensor because it is occluded by other targets or obstacles. Besides the already mentioned issues are all measurements delivered by a sensor subject to *random noise*. That makes the correct estimation of the target routes even more difficult [Cha11, HL01].

All in all simultaneously tracking multiple targets is a non-trivial task and it needs sophisticated probabilistic models and algorithms to tackle the *multiple target tracking* (MTT) problem [Cha11, Rei79].

1.2 Fundamental Tracking Approaches

In multiple target tracking a distinction is made between two fundamentally different tracking approaches.

Recursive tracking algorithms update their target estimations iteratively. In each time-step the new information of the current scan is incorporated into the track estimation

likelihoods[2] and a new estimation is calculated. Almost all recursive tracking algorithms are based on *Bayesian filtering theory*. Examples for recursive tracking algorithms are *Global Nearest Neighbor Tracking* [Bla86], *(Joint) Probabilistic Data Association* [BSJ72, FBSS83] and *Multiple Hypothesis Tracking* [Rei79].

Global tracking algorithms do not only consider the actual time-step in their update routine but a series of multiple time-steps at the same time. Hence, multiple measurements of consecutive time-steps are considered in relation and not only the measurements of the current time-step. Examples for global tracking algorithms can be found in [LSG07, ZLN08, HWN08].

1.3 Problem Formulation

In this master thesis a framework to tackle the MTT problem is implemented, analyzed and evaluated. It is required that the implementation is real-time capable even in tracking scenarios that contain a lot of targets and a lot of clutter. Additionally, the tracking framework should be easily adaptable to different tracking problems (e. g. tracking planes or people, considering acceleration and velocity, etc.) that it is versatile and universally applicable.
The recursive *Multiple Hypothesis Tracking* (MHT) algorithm was chosen to tackle the MTT problem. This choice was determined by the generally positive approach of scientists to the method as the preferred one for solving the *data association problem* in MTT systems [Bla04]. The term *data association* in that context refers to the difficulty of assigning the correct measurements within a scan to the different targets during the tracking process.

1.4 Thesis Structure

This master thesis is divided into four parts. The first part contains an introduction into single and multiple target tracking. The second part contains all mathematical theory be-

[2]The term *likelihood* refers to a value which measures how likely an event occurrence is. It can be expressed, for instance, as a probability.

hind multi hypothesis tracking. The third part covers the implementation and discusses difficulties and pitfalls in the actual implementation. The fourth part is the evaluation. More information about the different chapters can be found hereafter:

In chapter 2 the fundamental terminology and the concepts of single target tracking are presented. *State estimation* with the use of the *Kalman filter* and *gating* as important tracking tools are discussed. Finally, a few well-known single target tracking approaches are outlined.
In chapter 3 the concepts of multiple target tracking are presented. It is illustrated why tracking multiple targets simultaneously is such a difficult problem. Lastly, some well-known multiple target tracking approaches are outlined.
Chapter 4 constitutes the theoretical main part of this thesis and copes with the theory behind a real-time capable, state-of-the-art multiple hypothesis tracker. After the original version of the algorithm is introduced, different optimizations are discussed to speedup the algorithm.
Chapter 5 deals with the practical MHT implementation which was created for the purpose of this thesis. Difficulties and pitfalls that emerged during implementation and the chosen solutions are presented.
In chapter 6 the MHT implementation is evaluated. For that purpose the concepts behind multiple target tracking *performance measurement* are outlined and the *CLEAR MOT* performance metrics are presented. After that, the practical evaluation consisting of synthetic and real-world scenarios is performed.
In the last chapter 7 of this thesis an outlook of possible future improvements and a summary of what was achieved is given.

2 Single Target Tracking

In the field of target tracking, *single* and *multiple* target tracking techniques are distinguished as they differ largely in complexity. Hence, before multiple target tracking is tackled, the case of tracking only a single target is discussed. It lays down some fundamental ideas and techniques applied to all kinds of tracking problems.

2.1 Introduction

As stated in the introduction target tracking refers to the problem of following one or multiple targets over time. A target is any object which *state* (e. g. position and velocity) is of interest and thus is subject to tracking. The state of a target is measured by a sensor at instances of time. These instances of time are called *scans*. Each scan contains a certain number of *measurements* which describe the states of the present targets at that particular instance of time. It is assumed often that no two measurements of the same scan can originate from the same target. This means that each target in every scan generates at most one measurement. The number of measurements present in a scan depends on various factors such as the number of targets present in the scan volume, the *detection probability* or the amount of *clutter*[1].
In each time-step every measurement of the actual scan has to be classified to be either a false alarm, the start of a new *track* or the continuation of an existing track. This process is called *data association*. It is the crucial and most difficult part of every tracking algorithm. A *track* is a sequence of measurements that are hypothesized by the tracker to originate from the same single target.

[1] In the context of tracking, the term *clutter* refers to unwanted measurements contained in a scan. Clutter originates from the inability of the sensor to exclude disturbed signals or from faulty identified objects.

Important aspects of every tracking problem are, first of all, to distinguish between the single and multiple target cases. In the former case only one target can be tracked at the same time. In the latter case it is possible to track an arbitrary number of targets at the same time. Though, dependent on the used algorithm the number of targets has to be known a priori or can change dynamically as targets appear and disappear. Other aspects that have to be taken into account are the characteristics of the tracking environment and the sensor. High and low target density environments, as well as high and low clutter scenarios are distinguished. The ability of the sensor to detect a target is of importance as well. These properties are usually described by the *detection probability* $P_D \in [0,1]$, the density of *false alarms* β_{FT} which dependents on the *clutter rate* λ and the density of *new targets* β_{NT}. Algorithmic aspects of interest are the ability of correcting wrongly made associations, called *multiple-scan correlation* and the ability to cope with newly appearing and disappearing targets, called *track initiation* and *track termination* [Rei79].

In relation to the number of new measurements contained in a scan and the number of targets to track, different classes of algorithms were developed. In the simplest form of *single target tracking* (STT) only one target exists at the same time and each scan contains at most one new measurement. In that case the data association problem is trivial because it is determined by the underlying problem structure which track is associated with the new measurement. In that case the only improvement that can be made is the application of a *filter* to reduce noise in the potentially disturbed measurements. However, in practice simple tracking scenarios like this are very rare because the existence of clutter, causing scans with more than one measurement, is not considered.

The problem of tracking only a single target gets more complicated as soon as the underlying sensor delivers more than one measurement per scan. In that case, in each scan, a data association problem has to be solved because it has to be decided, which of the new measurements is assigned to the track. The additional measurements can originate e. g. from clutter or wrong assumptions about the number of targets in the scan volume.
In the case of scans containing multiple measurements it is important to *predict* the state of the track for the next time-step in order to find out which of the new measurements should be chosen to continue the track (e. g. by taking the measurement with the minimum distance to the state prediction). *State prediction* is crucial because no assumption about any future track state can be made without it and thus, assigning the next measurement is infeasible (especially in situations where intermediate measurements belonging

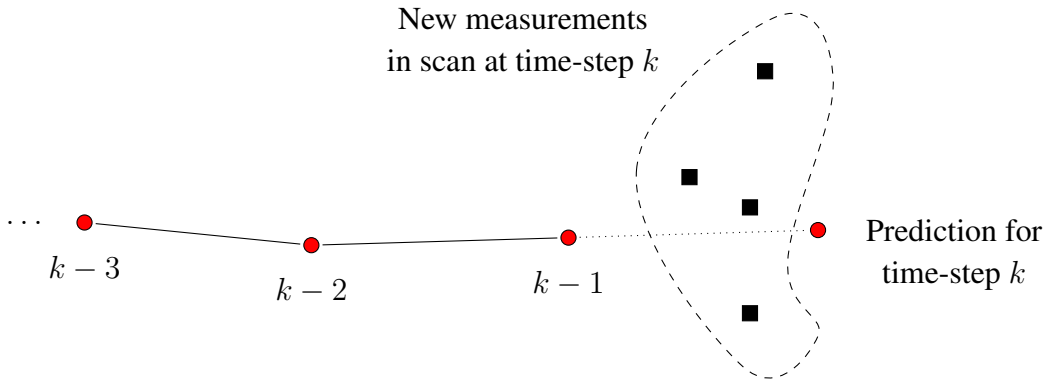


Figure 2.1: An illustration of the track state prediction process. A scan containing four new measurements arrived. Based on the track state prediction and a distance metric it can be decided which measurement to choose best to continue the track.

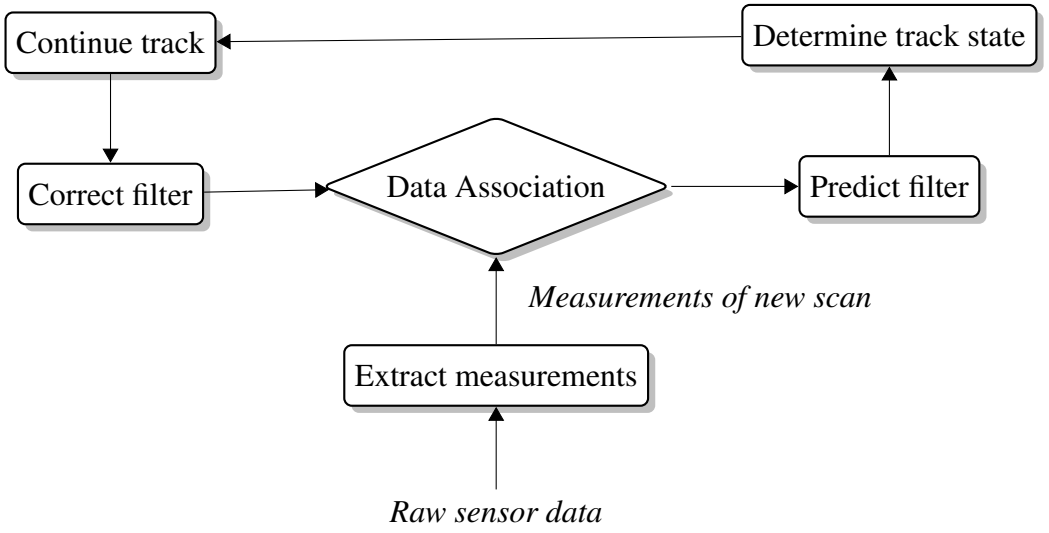


Figure 2.2: Fundamental structure of a STT system

to a track where omitted due to e. g. occlusion). In figure 2.1 the track state prediction process is illustrated.

Practically, in almost all tracking algorithms each track is associated with a filter which is used to predict the state(s) of the track for the next time-step(s) and which removes noise from the measurements. The general layout of a STT system is illustrated in figure 2.2. The most popular method for track state prediction in the context of tracking is the *Kalman filter*. It is described in conjunction with *Bayesian filters* in the following section.

2.2 State Space Estimation

As stated in the previous section, the prediction of the state of a target moving through the scan volume is essential for the data association process of recursive tracking algorithms. Therefore, the mathematical fundamentals of the *state space estimation* process are presented in this section. First, the general idea behind recursive *Bayesian filters* are outlined. After that the *Kalman filter* is presented which is the most popular representative of Bayesian filters for discrete data [HL01].

2.2.1 Recursive Bayes Filtering

A *Bayes filter* is a *recursive* filter. Based on a sequence of k measurements $z_1, \ldots, z_k$ made on a system, it computes the distribution $p(x_k|z_1, \ldots, z_k)$ of the system state x_k, given the prior initial distribution $p(x_0)$ and the *State Space Models*[2]

$$x_k \sim p(x_k|x_{k-1}) \text{ and}$$
$$z_k \sim p(z_k|x_k).$$

The filter computation is based on a recursive update rule which incorporates each new measurement z_k into the posterior distribution time-step by time-step .

It is assumed that the underlying stochastic process of the system system has the *Markov property*[3] and that the stochastic model is a first order *Hidden Markov Model*[4]. This means that any state x_k depends exclusively on the previous state x_{k-1}. The Markov assumption states that in any time-step k the past and future states are independent if the current state x_k is known [RAG04]. Formally this can be stated as

$$p(x_{k+1}|x_1, \ldots, x_k) = p(x_{k+1}|x_k).$$

[2] A *State Space Model* is one form to describe a dynamic physical system by using a set of input, output and state variables [Har04].

[3] The *Markov property* states that given the state of a system at a certain time, the future states are independent of its past [MS04].

[4] A *Hidden Markov Model* is a stochastic model which describes a system whose state is not directly observable, but only measurements on it can be obtained. The state of the system follows a *Markov process* [MS04].

Using *Bayes' theorem* $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ and the abbreviation $x_1, \ldots, x_n = x_{1:n}$, respective $z_1, \ldots, z_n = z_{1:n}$ with $n \in \mathbb{N}$, an expression for $p(x_k|z_1, \ldots, z_k)$ can be obtained:

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k, z_{1:k-1})p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})}$$
$$= \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{\int p(z_k|x_k)p(x_k|z_{1:k-1})\, dx_k}. \tag{2.1}$$

This is the *correction* step of the filter. Based on the prior distribution of equation (2.1) and the measurement likelihood $p(z_k|x_k)$, an equation for predicting the next system state can be derived. The derivation goes beyond the scope of this thesis and can be found e. g. in [Jaz70]. The final result is:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})\, dx_{k-1}. \tag{2.2}$$

This is the *prediction* step of the filter. The used relation is known as *Chapman-Kolmogorov* equation. It propagates the posterior distribution of the state from time-step $k-1$ to time-step k [RAG04]. In the next chapter one famous realization of the Bayes filter is presented.

2.2.2 Kalman Filter

The *Kalman filter* is the most popular representative of a *discrete* Bayesian filter. Thanks to its discrete nature, the integrals in equations (2.1) and (2.2) turn into sums. Hence, the recursive filter equations can be implemented directly using simple arithmetic and no numerical integration is required.

2.2.2.1 Introduction

The *Kalman filter* (1960) was developed by Rudolph E. Kálmán [Kal60]. It is a recursive filter used to predict the state of a *dynamic system* on the basis of a sequence of *disturbed* measurements, by minimizing the *root mean square* error. The Kalman filter is one of the most well-known and the most often used filtering algorithms from the toolbox of stochastic estimation methods. Essentially, it breaks down to a set of mathematical equations which implement a *predictor-corrector* type estimator. It is *optimal* in the sense that it minimizes the error covariance of the state of the system, provided that some presumed conditions are fulfilled [WB06].

The state of a system $x_k \in \mathbb{R}^n$ at time-step $k \in \mathbb{N}_0$ of a *time-discrete* process can be described by the following linear stochastic difference equation:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

The matrix $A \in \mathbb{R}^{n \times n}$ maps the state x_{k-1} of the previous time-step to the state of the current time-step k. The matrix $B \in \mathbb{R}^{n \times l}$ maps the *control input* $u \in \mathbb{R}^l$ from the previous time-step to the current one. The control input is optional and does not exist in many real-world problems. The system state is influenced by *process noise* described by the normally distributed random variable $w_k \in \mathbb{R}^n \sim \mathcal{N}_n(0,Q)$, with the process noise covariance matrix $Q \in \mathbb{R}^{n \times n}$. More information about the normal distribution is given in section A.1.

The connection between the system state x_k and a measurement $z_k \in \mathbb{R}^m$ at time-step k is given by the equation

$$z_k = Hx_k + v_k.$$

The measurement matrix $H \in \mathbb{R}^{m \times n}$ maps the system state x_k from *state space* to *measurement space*. Analogously to the state, the measurement is also subject to noise, the *measurement noise*. It is described by the normally distributed random variable $v_k \in \mathbb{R}^m \sim \mathcal{N}_m(0,R)$, with measurement noise covariance matrix $R \in \mathbb{R}^{m \times m}$. It is assumed that the measurement and the process noise described by the random variables v_k and w_k are *statistically independent* (uncorrelated).

In practice, the covariance matrices Q and R might change with successive time-steps and new measurements respectively. Nevertheless, in this introduction they are assumed to be constant.

2.2.2.2 The Kalman Equations

Let $\hat{x}_k^- \in \mathbb{R}^n$ be the *a priori* state prediction at time-step k, before the new measurement z_k is taken into account and $\hat{x}_k^+ \in \mathbb{R}^n$ the *a posteriori* state prediction at time-step k, after the new measurement z_k was taken into account. The a priori and a posteriori *prediction*

errors are defined as

$$e_k^- := x_k - \hat{x}_k^- \text{ and} \tag{2.3}$$

$$e_k^+ := x_k - \hat{x}_k^+. \tag{2.4}$$

Considering the zero-mean of e_k^- and e_k^+, the a priori and a posteriori prediction error covariance matrices[5] P_k^- and P_k^+ can be written in terms of the *expected value* $\mathrm{E}(\cdot)$[6] of the a priori and a posteriori prediction errors:

$$P_k^- = \mathrm{Cov}(e_k^-) = \mathrm{E}\left(e_k^- {e_k^-}^T\right) \text{ and} \tag{2.5}$$

$$P_k^+ = \mathrm{Cov}(e_k^+) = \mathrm{E}\left(e_k^+ {e_k^+}^T\right). \tag{2.6}$$

The next step in the derivation of the Kalman filter equations is to find an equation for calculating the a posteriori state prediction $\hat{x}_k^+$. The state prediction can be formulated as a linear combination between the a priori state prediction $\hat{x}_k^-$ and a weighted difference between the current measurement z_k and the measurement prediction $H\hat{x}_k^-$:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(\underbrace{z_k - H\hat{x}_k^-}_{=:r_k \in \mathbb{R}^m}),\ K_k \in \mathbb{R}^{n \times m}. \tag{2.7}$$

Thus, every new state prediction consists of a predictable fraction taken from the previous state that does not contain any new information and of a fraction that contains new information extracted from the new measurement. The difference vector $r_k = z_k - H\hat{x}_k^-$ is often referred to as the measurement *innovation* or *residual*. Its norm $\|r_k\|_2 = \sqrt{{r_k}^T r_k}$ is called *residuum*. The covariance matrix $S_k \in \mathbb{R}^{m \times m}$ of the innovation r_k is given below:

$$S_k := \mathrm{Cov}(r_k) = \mathrm{E}(r_k r_k^T) = \mathrm{E}((z_k - H\hat{x}_k^-)(z_k - H\hat{x}_k^-)^T) = HP_k^-H^T + R. \tag{2.8}$$

The matrix $K_k \in \mathbb{R}^{n \times m}$ is called *gain* or *blending factor*. It is chosen in a way that it minimizes the a posteriori error covariance of equation (2.6). This minimization can be accomplished by, first, substituting equation (2.7) into (2.4) and afterwards substituting the result of that into equation (2.6). Then the derivative of the trace with respect to K_k

[5] The *covariance matrix* of a vector $X = (X_1, \ldots, X_n)^T$ of n random variables is defined as $\mathrm{Cov}(X) = (\mathrm{Cov}(X_i, X_j))_{i,j=1,\ldots,n} \in \mathbb{R}^{n \times n}$.

[6] The connection between expected value and covariance is given by the equation $\mathrm{Cov}(X,Y) = \mathrm{E}(X,Y) - \mathrm{E}(X)\,\mathrm{E}(Y)$.

is taken. The result is set equal to zero and then solved for K_k. More details of this derivation can be found e. g. in [May79]. Consequently, the equation obtained finally is

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} = \frac{P_k^- H^T}{H P_k^- H^T + R} = \frac{P_k^- H^T}{S_k}. \tag{2.9}$$

By carefully looking at equation (2.9) it can be seen that as the measurement noise covariance R approaches zero, the innovation in equation (2.7) is weighted stronger: $\lim_{R \to 0} K_k = H^{-1}$. Contrarily, the innovation is weighted less strongly as the a priori estimate error covariance P_k^- approaches zero: $\lim_{P_k^- \to 0} K_k = 0$.
More intuitively this behavior can be explained by considering that as the measurement noise covariance R gets smaller, more trust can be put into the correctness of the actual measurement z_k while the predicted measurement $H\hat{x}_k^-$ is trusted less. The same holds the other way around for an increasing measurement error covariance.

2.2.2.3 Filtering Algorithm

The Kalman filter is applied as an alternating two-step process of predicting the new state (*prediction* step) and updating the filter with a new measurement (*correction* step). The prediction equations are responsible for projecting the current state and error covariance estimates forward in time in order to obtain the a priori estimate for the next time-step. The correction equations are responsible for improving the a priori estimate by incorporating the information of the new measurement and obtaining an improved a posteriori estimate. The process of predicting and correcting is illustrated in figure 2.3 together with the appropriate equations.

2.3 Gating

In high density tracking scenarios the number of measurements delivered in a scan is potentially big. In that case determining the new track state becomes easily computationally infeasible because too many measurements have to be considered per track. To overcome this problem a technique known as *gating* is employed. It reduces the number of measurements that can be possibly associated with a track. This is accomplished by defining a region in the scan volume for each track where measurements are allowed to originate from for that they are assigned to this particular track. The regions in question

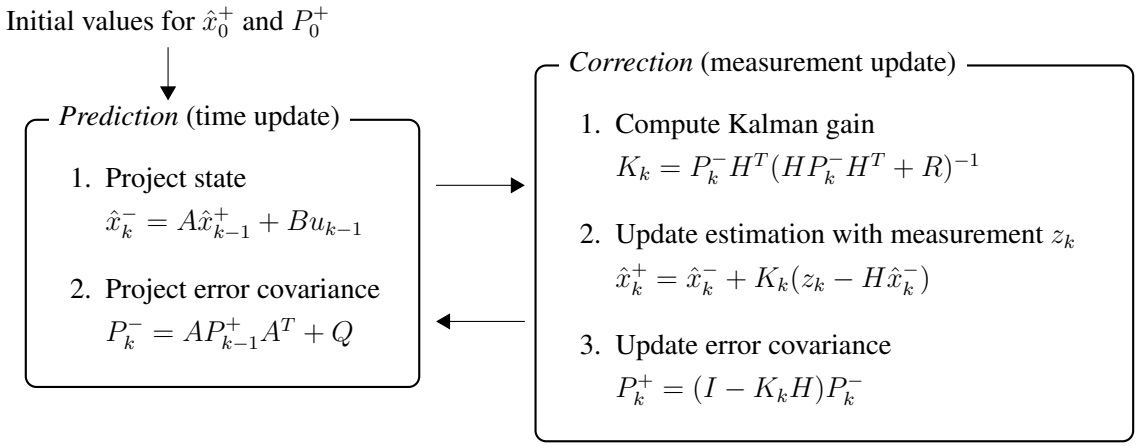


Figure 2.3: Summary of the recursive Kalman filter application [WB06].

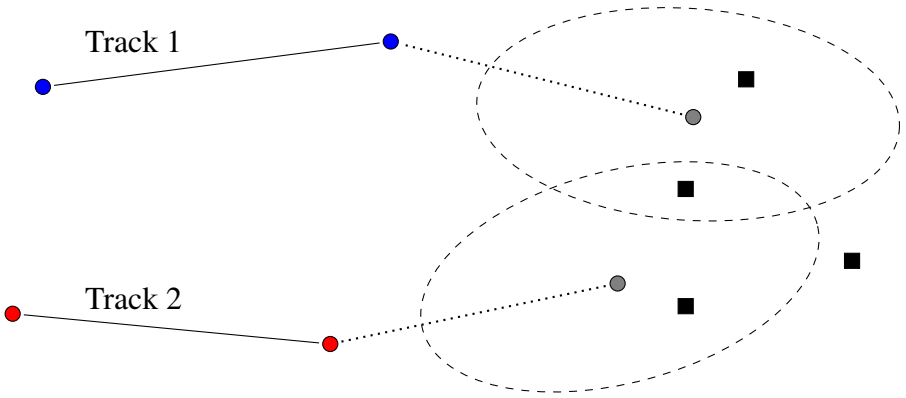


Figure 2.4: Tracks with their gates associated with the predicted states (gray circles) and some measurements (black squares) [HL01].

are called *gate* or *validation region*. In figure 2.4 two tracks with their gates associated with the predicted states for the next time-step and some measurements are depicted.

The shape and size of the gate of a track is defined by its associated Kalman innovation covariance matrix S_k from equation (2.8). Its center Hx_k is given by the state of the track projected into the measurement space. To determine whether a certain measurement falls into the gate of a track, the *Mahalanobis* distance (for more information see section A.1) between the predicted state of that track and the measurement in question is calculated. If the resulting distance is greater than a certain threshold $\eta \in \mathbb{R}^+$, this measurement is no longer considered for that track. Given the predicted state x_k,

the measurement z_k and the innovation covariance matrix S_k the gating condition can be formulated as

$$r_k{}^T S_k{}^{-1} r_k = (z_k - Hx_k)^T S_k{}^{-1}(z_k - Hx_k) \leq \eta, \tag{2.10}$$

It can be shown that the lefthand side of this equation follows a χ^2_n-distribution with $n = \dim(z_k)$ degrees of freedom (DOF) as the sum of normally distributed random variables (for a proof of this theorem see section A.1). Thus, the size η of the gate can be obtained on the basis of the probability $P_G \in [0,1]$ that a certain amount of measurements fall into the validation region. Using the inverse of the χ^2 cumulative distribution function (CDF) $F^{-1}_{\chi^2(n)}(x)$, the gate size η can be determined [Cox93]. It holds for the gating probability P_G that

$$P_G = F_{\chi^2(n)}(x) = P\left(\frac{n}{2}, \frac{x}{2}\right) = \frac{\gamma(\frac{n}{2}, \frac{x}{2})}{\Gamma(\frac{n}{2})},$$

where $\gamma(n,x)$ is the *lower incomplete* Gamma function and $P\left(\frac{n}{2}, \frac{x}{2}\right)$ is the *regularized* Gamma function (for more information see section A.1). It describes the probability that χ^2_n lies in the interval $[0,x]$. As the Gamma function cannot be solved analytically the values have to be calculated numerically. However, for $n = 2$ DOF the CDF is of the form $F_{\chi^2(2)}(x) = 1 - e^{-\frac{x}{2}}$, which can be inverted and solved analytically as $F^{-1}_{\chi^2(2)}(x) = -2\ln(1 - P_G)$.

For example for 2-dimensional measurements and a probability of 95% that true measurements fall into the gate, its size has to be set to $\eta = F^{-1}_{\chi^2(2)}(P_G = 0.95) \approx 5.99$. In figure 2.5 gating regions for different gating probabilities P_G are depicted.

It is important to mention that a validation region does not provide any statistical measure for the rejection of false alarms (clutter). It only defines an acceptance region in which $(100P_G)\%$ of the true measurements are accepted [BUDW06, Cox93].
The exclusion of measurements with low assignment probabilities results in an effective reduction of the combinatorial complexity of the data association problem.

2.4 Some STT Approaches

Several approaches to tackle the STT problem have been devised in literature over the last decades. In this section, two popular approaches of different complexity are outlined.

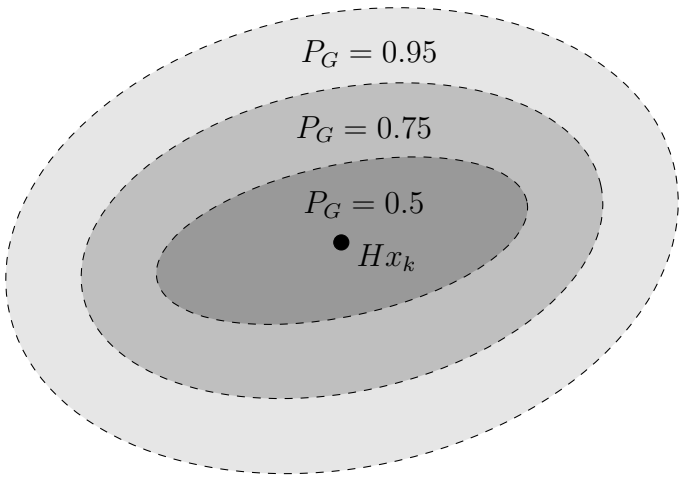


Figure 2.5: Gates for different gating probabilities P_G and constant innovation covariance matrix S_k. The gates are centered on the prediction projected into measurement space Hx_k.

2.4.1 Nearest Neighbor Standard Filter

The *Nearest Neighbor Standard Filter* (NNSF) is the simplest approach to STT [BSF87]. NNSF applies the following data association policy. First, all measurements which do not gate with the track are discarded. Second, the measurement which has the smallest distance to the predicted track state in terms of the Mahalanobis distance is chosen for track continuation.

The application of the NNSF requires a low clutter rate λ and a high detection probability P_D. In scenarios where targets are moving in close proximity or scenarios where the underlying sensor delivers a lot of clutter, there is the possibility of performing many wrong assignment decisions. In that context the biggest limitation of the NNSF is that it does not support multiple-scan correlation. Any measurement assignment performed in a time-step is irreversible, even though information obtained in later scans can be often of great value for resolving ambiguities occurring in the current scan.

Still, the NNSF is widely used because of its simplicity and its acceptable results in a range of simple tracking scenarios [Cox93]. Examples for the application of nearest neighbor tracking can be found in [CSD88, DF91].

2.4.2 Probabilistic Data Association Filter

The *Probabilistic Data Association Filter* (PDAF), first presented by Bar-Shalom and Tse [ST75], is an important representative of the *soft decision* class of tracking algorithms [BSJ72]. Algorithms of this class do not perform fixed measurement associations like the NNSF, but they calculate a new track state based on multiple measurements of a scan. The NNSF takes only one of the received measurements into account, associates it and discards the others. This approach works well in sparse target and low clutter scenarios but it is likely to fail in dense scenarios or scenarios with an increasing rate of false alarms. In that case the PDAF can improve tracking results significantly.

The PDAF continues the track with a *weighted average* of all measurements falling into the gating region. The weight of each measurement is calculated on the basis of the probability of associating the measurement in question to the track. Hence, the weight is called *association probability*. When calculating an association probability, the probability of the considered measurement being a false report is also taken into account. This enables the usage of the PDAF for target tracking in cluttered environments [BSDH09].

There exist two different PDAF versions. They differ in the underlying model which is assumed for the distribution of false alarms. The *parametric* version assumes a Poisson distribution, while the *non-parametric* version assumes a uniform distribution of false alarms [Cox93].

3 Multiple Target Tracking

3.1 Introduction

The most difficult tracking problems arise when more than one target is subject to tracking at the same time. Such *multiple target tracking* (MTT) scenarios are of great importance because they occur in a lot of fields of application. It turns out that the MTT problem is surprisingly hard in comparison to the single target case.

It could be assumed that when tracking a single target has a certain amount of computational costs, tracking n similar targets would just have n times this costs. However, the reality appears different. In fact, the complexity of the simplest MTT approaches is, at least, proportional to the square of the number of targets tracked, when assuming that in each scan n new measurements are delivered.
In MTT the data association problem constitutes the task of assigning multiple different measurements within each scan to multiple different tracks. When n targets are tracked simultaneously and the actual scan contains n new measurements, each of the n new measurements could in principle be assigned to any of the n tracks. The need of considering every single association possibility makes MTT a problem basically proportional to $\mathcal{O}(n^2)$ for n targets. Handling this algorithmic complexity is the first major difficulty of MTT [Uhl92, HL01].

The second major difficulty of MTT is not of algorithmic nature. It concerns the uncertainty caused by sensor inabilities like sporadic miss detections or the delivery of false alarms. In figure 3.1 the increased difficulty of MTT in comparison to STT is illustrated.

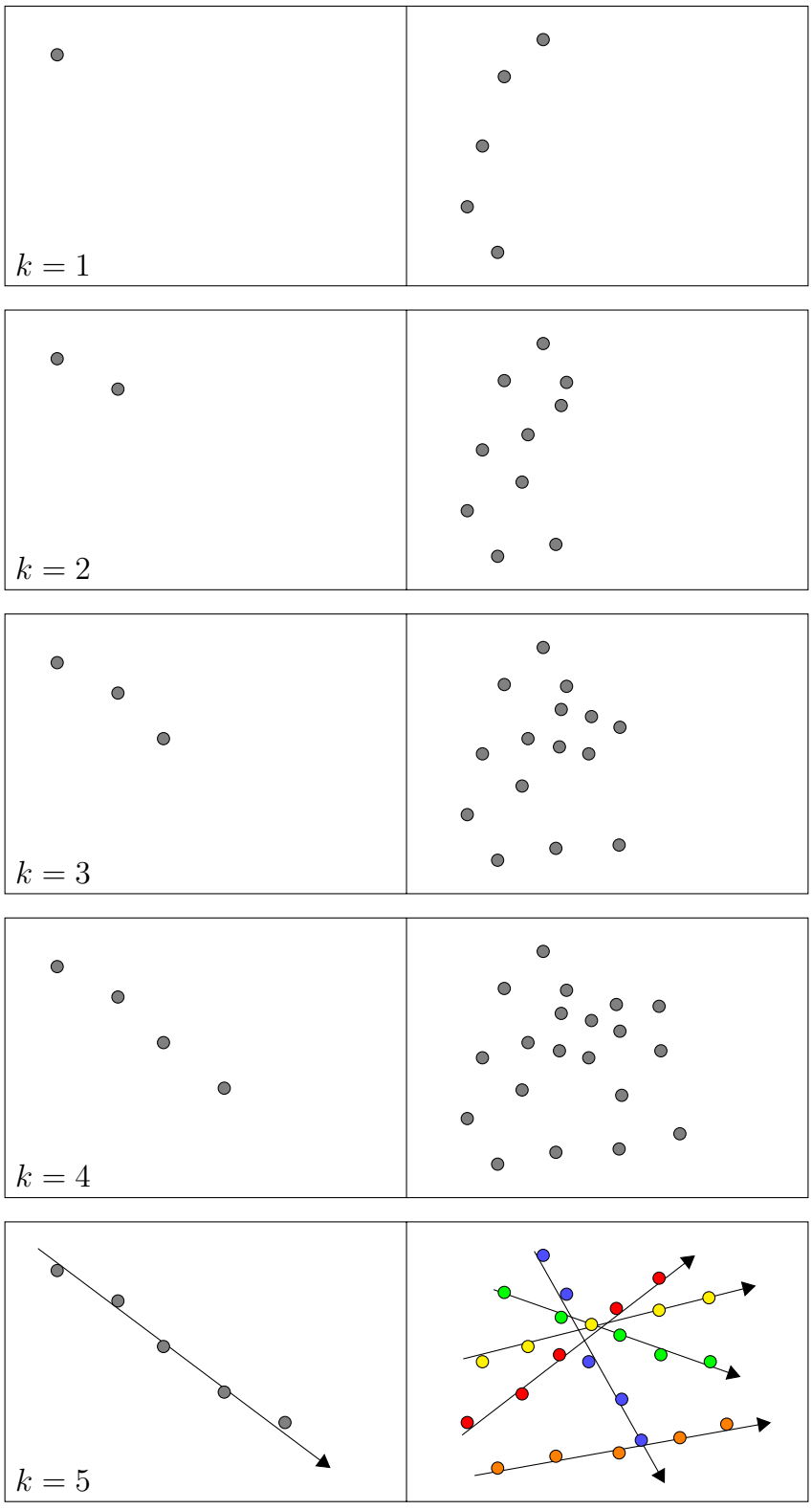


Figure 3.1: Comparison of STT (left) and MTT (right) [HL01].

3.2 Single Source Assumption

In many MTT algorithms it is assumed that every measurement originates at most from one source: being a false target, a new target or an existing target. After a measurement was classified by the MTT algorithm to originate from one of the three mentioned sources, it is not possible anymore that the same measurement is associated with any other source [Rei79]. In the following it is referred to this property as *single source assumption*. MTT algorithms that fulfill the single source assumption only create *disjoint* tracks that do not share any common measurements. Disjoint tracks are also called *compatible*.
The fulfillment of the single source assumption is mostly of great importance for two reasons. On one hand, it reflects the underlying physical nature of the tracking problem. On the other hand, the combinatorial complexity of the data association problem would be even bigger without this constraint and thus, most of the efficient algorithmic approaches were unfeasible.
Almost all MTT algorithms fulfill the single source assumption but there are some exceptions like the *Track Splitting Filter* outlined in the following section.

3.3 Some MMT Approaches

Different approaches are devised in literature to tackle the MTT problem. In this section three of them are outlined. The last one, being *Multiple Hypothesis Tracking*, is described in full detail in the next chapter. The first two approaches are just extensions of their STT counterparts that were already presented in section 2.4.

3.3.1 Global Nearest Neighbor Tracking

Global Nearest Neighbor Tracking (GNNT) is an extension of the Nearest Neighbor Standard Filter (NNSF), described in section 2.4.1, to multiple targets [Bla86]. It is assumed that the number of targets in the scan volume is known beforehand and it does not change over time (track initiation is not supported).

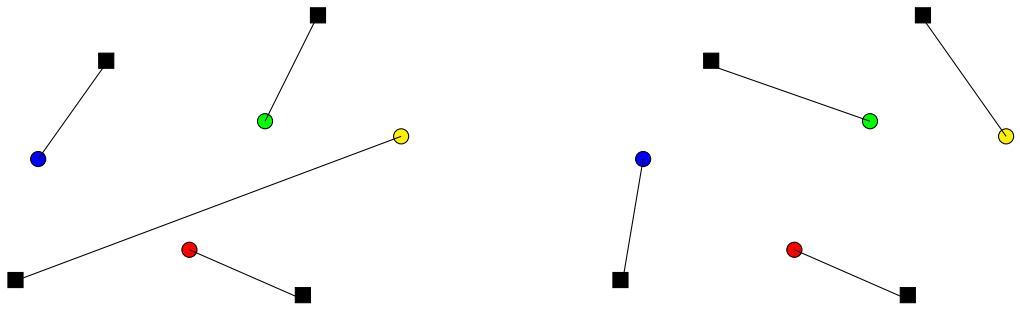


Figure 3.2: Comparison of greedy nearest neighbor association (left) and global minimization of distances using the GNNT approach (right).

GNNT fulfills the single source assumption. Thus, each measurement is allowed to be associated at most with one track. It is not clear which measurement should be associated best to which track in order to minimize the global overall distance. Any *greedy* approach that just assigns the closest measurement to each track might fail as exemplarily shown in figure 3.2. In order to minimize the global distance for all measurements, an assignment problem needs to be solved in each scan. To reduce the amount of measurements which have to be considered, GNNT also makes use of the aforementioned gating technique. Thus, only measurements falling into any gate are considered [KUS03].

3.3.2 Joint Probabilistic Data Association Filter

The *Joint Probabilistic Data Association Filter* (JPDAF) [FBSS83] is an extension of the original Probabilistic Data Association Filter (PDAF) [BSJ72], described in section 2.4, to the multiple target case. It can deal better with multiple targets in clutter and dense target scenarios than GNNT because it does not perform hard data associations [BSF87]. The algorithm requires that the number of targets to track is known beforehand and that it is constant (track initiation is not supported). The JPDAF fulfills the single source assumption.

The JPDAF calculates measurement-to-track association probabilities jointly across the tracks existing in the current time-step. Then each track is updated with a weighted innovation consisting of fractions of all measurements. The higher the association probability between a measurement and a track in question is, the bigger is the fraction of this particular measurement on the innovation for that track.

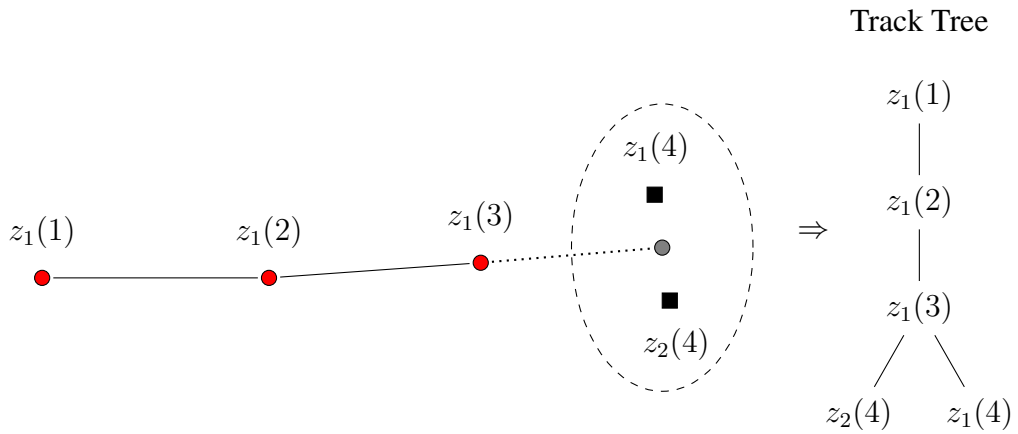


Figure 3.3: In time-step 4 two measurements $z_1(4)$ and $z_2(4)$ fall into the gate of the track. Accordingly, the TSF adds two branches to the track tree: one representing the assignment of $z_1(4)$ to the track and the other one the assignment of $z_2(4)$ [Cox93].

3.3.3 Track Splitting Filter

Static tracking algorithms like GNNT or the JPDAF perform data associations in a way that the decision at a time-step k is made by only considering measurements known up to that time-step. When applying such data association schemes each measurement-to-track-association is *irrevocable* once made. The result is that tracking algorithms using static data association schemes do not perform particularly well under high clutter or in dense target environments. This issue motivated the development of schemes like the *Track Splitting Filter* (TSF) [SB75] which is capable of reverting prior data association decisions that later turn out to be incorrect.

The TSF forms a tree of measurement-to-track associations which is called a *track tree*. In such a track tree each branch from the root to a leaf represents one *track hypothesis*. Whenever there is more than one measurement falling into the gate of a track, for each of these measurements a branch in the respective track tree is created. It is important to note that no *final* assignment decision is made at this stage of the algorithm. It is implicitly assumed that decision ambiguities at time-step k are resolved by future scans. In every time-step the most likely tracks are determined by calculating the probability for every track hypothesis and taking the most likely one from each track tree. In figure 3.3 the principles of track trees are illustrated.

Obviously, track trees can become quickly very large if there are many measurements falling into the same gating regions. Hence, it is crucial for the practicability of this approach to delete unlikely branches from the track trees. Unlikely branches are identified by their track probabilities.

One big drawback of the TSF is that it does not fulfill the single source assumption for measurements. Therefore, it can happen that the same measurement is assigned to two different tracks (track trees) [Cox93].

3.3.4 Multiple Hypothesis Tracking

One very successful approach for tackling the multiple target data association problem is known as *Multiple Hypothesis Tracking* (MHT) [Rei79]. Similar to the TSF, MHT constitutes a *deferred decision logic* which is capable of postponing difficult measurement-to-track association decisions to a future point in time when more information is received. Nowadays, scientists agree that MHT is the preferred method for tackling the problem of tracking multiple targets in cluttered and dense environments [DN06, Bla04].

The basic idea of MHT is to tolerate ambiguities arising during the data association process by maintaining multiple data association hypotheses. As a new set of measurements arrives MHT forms and evaluates all possible track-to-measurement associations, the so called *hypotheses*. All previously formed hypotheses are kept among multiple scans. This enables the algorithm to use later arriving measurements to aid in solving ambiguities of prior track-to-measurement associations.
However, storing multiple hypotheses at the same time comes at the price of increased memory requirements and computational costs. Maintaining a large number of hypotheses, while keeping runtime costs low, is an essential requirement of any practical MHT implementation. The inherent exponential complexity of Reid's original MHT formulation let to a number of optimization techniques, ranging from pruning strategies to reformulations of the algorithm [Rei79, Bla04].

In the next chapter the original MHT algorithm of Reid is discussed first and afterwards, some important optimization techniques are presented.

4 Multiple Hypothesis Tracking

4.1 Introduction

Track initiation, track termination and a deferred decision logic are important features of modern tracking systems. After all, it is rarely the case that all targets are known a priori and that they are present during the whole tracking process. Crucial for the application of the tracking algorithm in high clutter and high target density scenarios is its ability to revert faulty associations in the future as new information is received.

One approach that naturally supports these three important features is the *Multiple Hypothesis Tracking* (MHT) algorithm. It was already briefly described in section 3.3.4. The MHT algorithm was first described by Reid in his seminal paper from 1979 [Rei79]. The fundamental idea of MHT is to delay difficult data association decisions until more information is received. This requires maintaining a set of different association *hypotheses* in a ranked fashion in order to determine the most likely hypothesis within each scan. The inherent problem of MHT is that the number of hypotheses grows exponentially, which means a combinatorial explosion quickly exhausting memory and computation capabilities of every computer. Thus, an important component of every MHT system are techniques for keeping the number of hypotheses maintainable [Bla04].

In this chapter Reid's original formulation of the MHT algorithm (henceforth called *conceptional MHT*) with its fundamental hypothesis probability evaluation equation is first presented. Afterwards, different techniques for solving practical algorithm issues are discussed, including the fundamental reformulation optimization of Cox and Hingorani [CH96]. This reformulation makes the MHT approach, for the first time, applicable on arbitrary tracking scenarios.

4.2 Previous Work

The MHT algorithm dates back to Morefield's MTT *batch-processing* approach (1977) [Mor77]. Morefield's algorithm is based on a *0–1 integer linear programming*[1] problem. He formulated the MTT problem as a pattern recognition problem that classifies measurements based on Bayesian decision theory. This effectively associates the measurements in such a way to different tracks that they fit best a given trajectory model.

Reid (1979) took over Morefield's original ideas but fundamentally changed the structure of the algorithm to make it sequentially applicable [Rei79]. Reid's algorithm maintains multiple different association hypotheses while processing the set of measurements of each new scan. By applying different *pruning* strategies only a fixed number of the most likely hypotheses is kept. This yields a good approximation of the NP-hard multi-scan data association problem of Morefield. Reid's method is much better suited for real-time applications because the hypothesis set can be updated *iteratively* from scan-to-scan. Unfortunately, as the enumeration of all association hypotheses in each scan is required, the complexity of the approach grows exponential in computation time and memory. This makes the algorithm unfeasible in scenarios containing a big number of tracks and measurements [CM95].

Nagarajan et al. (1987) presented an algorithm in which the n-best hypotheses are found through an "easy search process", instead of an "extensive enumeration" [NCS87a, NCS87b]. However, the authors do not give any theoretical proof of their technique. Even though there is strong evidence given that their technique leads generally to a huge reduction in computation time, it could be shown that the worst-case is still of exponential complexity [CM95].

Brogan (1989) provided an algorithm which determines a ranked set of n assignment hypotheses [Bro89]. It avoids as well an exhaustive hypotheses enumeration. However, it is not guaranteed that this set really contains the n-best assignments. Thus, it might

[1] *Integer linear programming* (programming $\triangleq$ optimization) is a field of applied mathematics which belongs to the domain of linear optimization. Like linear optimization it is about finding a way to obtain the best possible output in a given mathematical model, the so called *objective function*, described as a list of linear relationships. However, all linear equations may only contain integer coefficients or for 0–1 integer linear programming only 0 or 1.

happen that some good association hypotheses are missed. Still, there is a condition given of how to determine n so that, at least, the first $m \leq n$ assignment hypotheses are optimal. Unfortunately, again no theoretical analysis of the runtime complexity is provided in the literature [CM95].

Danchick and Newnam (1993) presented for the first time an algorithm for determining exactly the n-best hypotheses without requiring an exhaustive enumeration [DN93]. They recognized first that it is possible to determine the best data association hypotheses by reformulating Reid's original enumeration procedure as a classical *linear assignment problem* (LAP). They showed how modifications to the cost matrix and the repeated application of a LAP solver, like the *Hungarian method*, leads to the n-best assignment hypotheses. Nevertheless, their approach has two major disadvantages. Primarily, in the worst-case $n!$ LAPs have to be solved (even though the average case is expected to be better). Second, after each iteration duplicate assignments have to be identified and removed.
Cox and Miller (1995) found a better solution to identify the exact n-best hypotheses [CM95]. They exploited *Murty's algorithm* from the field of *operations research*[2] to determine the ranked set of best association hypotheses [Mur68]. The major advantage of Danchick and Newnam's approach is that the number of LAPs to be solved is linear in n instead of $n!$. Furthermore, no duplicate LAPs are solved and thus the removal of duplicate hypotheses is not needed. A mathematically well-founded derivation of this technique can be found in [DN06].

Murty's algorithm (1968) for determining the n-best data association hypotheses is based on an algorithm for solving LAPs (e. g. the Hungarian method [Kuh55] or the Munkres algorithm [Mun57]). Jonker and Volgenant (1987) created an optimized algorithm for solving square LAPs [JV87]. The algorithm outperforms classical methods for dense LAPs easily [PPBS99]. Bijsterbosch and Volgenant further improved this algorithm and extended it to the rectangular case [BV10]. Today, it is the most widely used algorithm for solving the data association problem in MTT. Miller et al. (1997) proposed different optimizations to Murty's algorithm in conjunction with Jonker and Volgenant's LAP solver in the context of MTT which improved the performance considerably [MS+97].

[2] *Operations research* is an interdisciplinary, mathematical science focusing on the effective use of technology by organizations. Techniques from different mathematical science, including optimization are employed.

4.3 Conceptional Multiple Hypothesis Tracking

In this section Reid's original MHT algorithm is presented. First, elementary principles of his approach are introduced. Afterwards, the hypothesis generation algorithm is presented. Finally, an equation to calculate the likelihood of a hypothesis is derived. The equation constitutes the main achievement of this section taking into account its importance for the selection of the most likely n hypotheses within each time-step.

4.3.1 Algorithm Overview

Before a formalized formulation of the MHT algorithm is provided, a more intuitive description based on an example is given.
In figure 4.1 an ambiguous and thus, non-trivial data association situation is depicted. The validation regions of the two tracks overlap. The two measurements M_2 and M_3

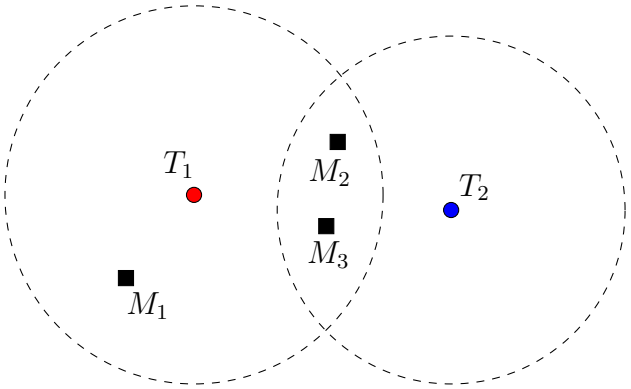


Figure 4.1: An example for a data association conflict [Bla04]

gate with both of them, whereas measurement M_1 only gates with the first track. It is assumed that these two tracks T_1 and T_2 are represented by a hypothesis H at time-step $k-1$, prior to the arrival of the new measurements M_1, M_2 and M_3 in time-step k.
For example a likely hypothesis would be that the measurement M_2 belongs to the track T_1, the measurement M_3 to the track T_2 and that the last measurement M_1 is the start of a new track T_3. Another unlikely hypothesis would be that all three new measurements originate from clutter and thus they are classified as false alarms; neither updating track T_1 and T_2 nor initiating any new track. Even for simple tracking problems, as the one of the example, the number of feasible hypotheses is large. In table 4.1 exemplarily five

Number	Hypothesis
1	$M_1 \to T_1$, $M_2 \to T_2$, $M_3 \to NT$
2	$M_1 \to T_1$, $M_2 \to T_2$, $M_3 \to FT$
3	$M_1 \to T_1$, $M_2 \to T_3$, $M_2 \to NT$
4	$M_1 \to T_1$, $M_2 \to T_3$, $M_2 \to FT$
5	$M_1 \to NT$, $M_2 \to T_1$, $M_3 \to T_2$
⋮	⋮

Table 4.1: Five feasible posterior data association hypotheses for H, where NT means a new target and FT means a false alarm [Bla04].

feasible posterior data association hypotheses for H are listed.

The MHT algorithm generates all feasible data association hypotheses for arbitrary tracking scenarios in a systematic fashion. In figure 4.2 the basic structure of the MHT algorithm is depicted.
Each iteration at a time-step k begins with the set of old hypotheses from the previous iteration at time-step $k-1$. Each of these old hypotheses becomes a parent hypothesis for the current time-step k as new hypotheses are formed. The MHT algorithm fulfills the single source assumption (see section 3.2). Hence, a hypothesis provides an interpretation of all previously received measurements as a set of disjoint tracks. Before the actual data association takes place, the Kalman filter associated with each track is predicted. This results in the estimation of the track states for the upcoming time-step. Next, the newly arrived measurements are matched with the track state predictions using the Mahalanobis distance. The matching process knows three different possible outcomes for each measurement association. Every measurement can be:

1. the continuation of a previously known track,
2. the start of a new track,
3. a false alarm due to clutter, or other sensor inabilities.

The MHT algorithm enumerates all possible data association hypotheses effectively generating a tree. That tree encodes all association hypotheses as branches from the root to one of its leaves. The tree depth increases with every processed measurement. For each hypothesis a likelihood for assigning the respective measurement is calculated. The probability stands for the likelihood of a hypothesis and is required to rank the hypotheses

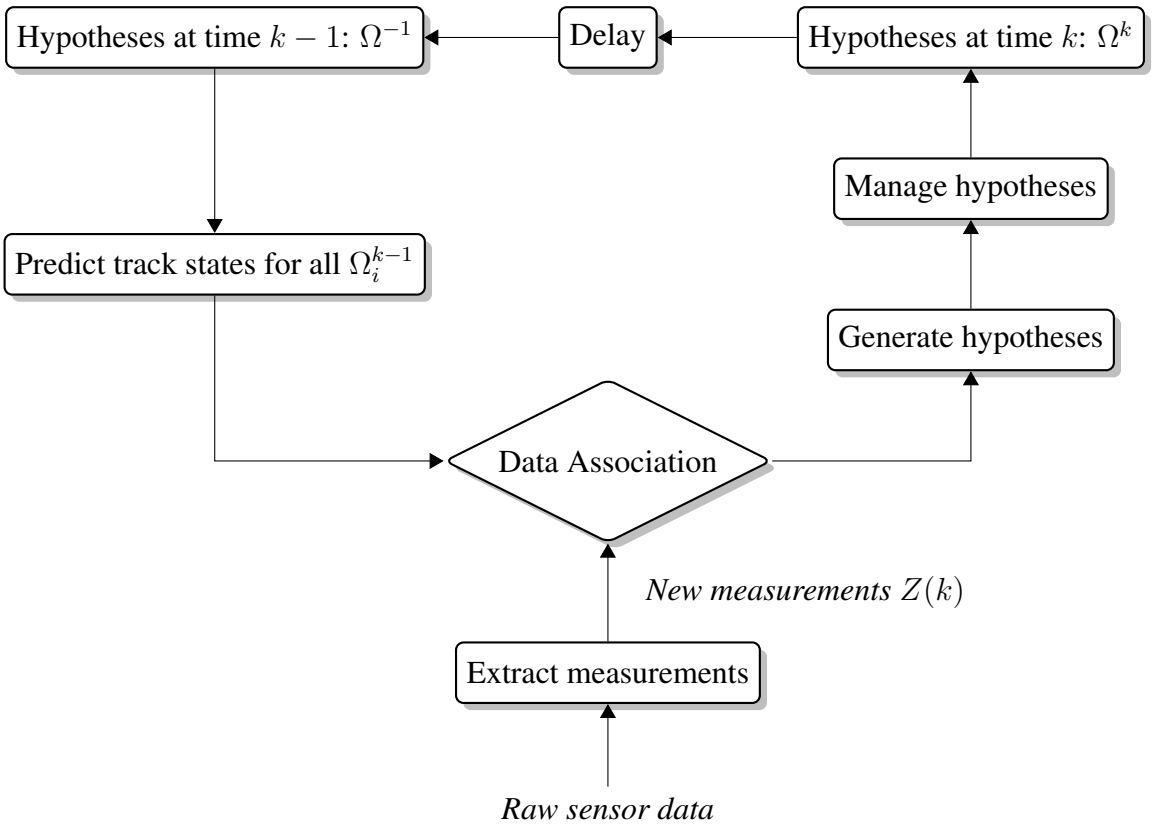


Figure 4.2: Fundamental structure of the MHT algorithm [CH96]

from the most likely to the most unlikely. Finally, the hypothesis tree is pruned and unlikely hypotheses are removed to keep the size of the tree in check.

4.3.2 Hypothesis Generation

The MHT algorithm generates sets of new hypotheses in an iterative manner on the basis of the set of hypotheses from the previous scan. As already mentioned, the MHT algorithm fulfills the single source assumption. Therefore, the hypotheses generation method has to control that no two measurements of the same scan are associated with the same target. Thus, no two tracks of the same hypothesis have a measurement in common. They are all mutually compatible.

Measurements originating from a sensor are modeled as D_M-dimensional vectors $z \in$

$\mathbb{R}^{D_M}$. The states maintained by the Kalman filter are modeled as D_S-dimensional vectors $x \in \mathbb{R}^{D_S}$. The set of M_k measurements delivered in scan k is denoted by

$$Z(k) := \{z_i(k), i = 1, \ldots, M_k\} \subset \mathbb{R}^{D_M}. \tag{4.1}$$

The set of all scans up to time-step k is denoted by

$$Z^k := \bigcup_{i=1}^{k} Z(i). \tag{4.2}$$

The set of all H_k hypotheses at the time of scan k, associating the set of measurements Z^k up to scan k with tracks or clutter, is denoted by

$$\Omega^k := \{\Omega_i^k, i = 1, \ldots, H_k\}. \tag{4.3}$$

An association hypothesis Ω_i^k is a 5-tuple consisting of a set of tracks describing the measurement-to-track associations, the number of tracks in this hypothesis N_{TGT}, the number of measurements in the actual scan associated with previously existing tracks N_{DT}, the number of measurements in the actual scan that initiated new targets N_{NT} and the number of measurements in the actual scan classified as false alarms N_{FT}. Their sum is equal to the number of measurements $M_k = N_{DT} + N_{FT} + N_{NT}$. The meaning of these counts is illustrated in figure 4.3.

When in scan $k + 1$ a new set of measurements $Z(k + 1)$ is received, a new set of hypotheses Ω^{k+1} is formed iteratively as follows.
Let $\hat{\Omega}^m$ denote the set of hypotheses after the m^{th} measurement $z_m(k + 1)$ of the actual scan is processed. The process starts by initializing $\hat{\Omega}^0 = \Omega^k$. Then a new set of hypotheses $\hat{\Omega}^m$ for each prior hypothesis $\hat{\Omega}_i^{m-1}$ and each new measurement $z_m(k + 1)$ is formed. Each hypothesis in this new set is the joint hypothesis which claims that $\hat{\Omega}_i^{m-1}$ is true and that the measurement $z_m(k + 1)$ is either a false alarm, the beginning of a new target or the continuation of one of the prior targets. This procedure is repeated for every measurement until the set of hypotheses $\Omega^{k+1} = \hat{\Omega}^{M_k}$ is formed. The hypothesis generation process is outlined in pseudo-code in algorithm 4.1.

The iterative hypothesis generation method is well suited to be be implemented with the use of a tree data-structure. In such a tree each branch from the root to one of its leaves represents a different data association hypothesis. Each node describes the assignment of one measurement to an existing track, a false target or a new target. In figure 4.4

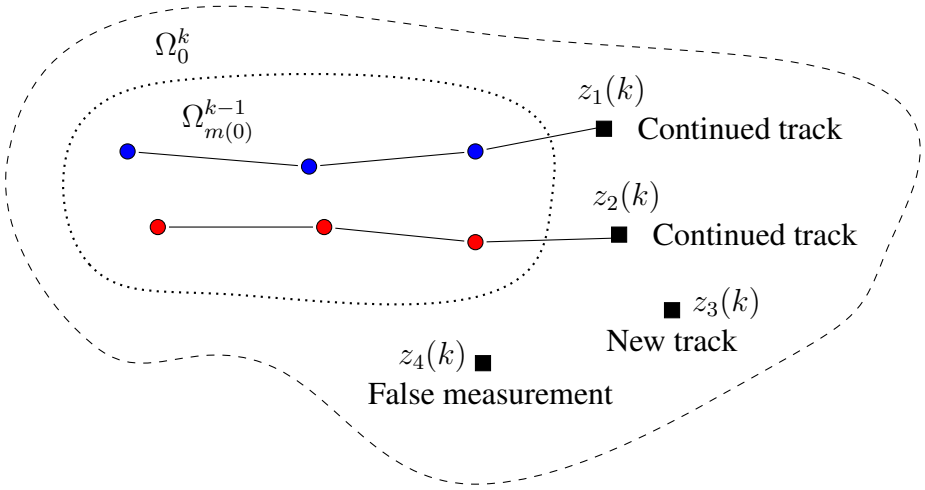


Figure 4.3: The initial hypothesis $\Omega_{m(0)}^{k-1}$ contains $N_{TGT} = 2$ tracks, each of them consisting of 3 measurements. In time-step k a new scan $Z(k) = \{z_1(k),z_2(k),z_3(k),z_4(k)\}$ with $M_k = 4$ measurements arrives. A new hypothesis Ω_0^k is formed which associates $N_{DT} = 2$ out of 4 measurements to the 2 existing tracks. $N_{NT} = 1$ measurement initiates a new track and $N_{FT} = 1$ measurement is classified as false alarm.

Algorithm 4.1 Reid's hypothesis generation method

Input: Set of hypotheses Ω^k and set of measurements $Z(k+1)$

Output: Set of hypotheses Ω^{k+1}

1: $\hat{\Omega}^0 = \Omega^k$
2: **for** $z_m(k) \in Z(k+1)$ with $m = 1,\ldots,M_k$ **do**
3: $\quad\hat{\Omega}^m = \emptyset$
4: $\quad$**for** $\hat{\Omega}_i^{m-1} \in \hat{\Omega}^{m-1}$ **do**
5: $\quad\quad\hat{\Omega}^m = \hat{\Omega}^m \cup \{\hat{\Omega}_i^{m-1} + z_m(k)$ as false target$\}$
6: $\quad\quad\hat{\Omega}^m = \hat{\Omega}^m \cup \{\hat{\Omega}_i^{m-1} + z_m(k)$ as new target$\}$
7: $\quad\quad$**for** tracks $t \in \hat{\Omega}_i^{m-1}$ **do**
8: $\quad\quad\quad$**if** $z_m(k)$ gates with track t **and** t has not been continued yet **then**
9: $\quad\quad\quad\quad\hat{\Omega}^m = \hat{\Omega}^m \cup \{\hat{\Omega}_i^{m-1} + z_m(k)$ continuing track $t\}$
10: $\quad\quad\quad$**end if**
11: $\quad\quad$**end for**
12: $\quad$**end for**
13: **end for**
14: **return** $\Omega^{k+1} = \hat{\Omega}^{M_k}$

a tracking scenario and its corresponding *hypothesis tree* is depicted. This approach is known as *hypothesis-oriented* MHT.

4.3.3 Probability of a Hypothesis

Rating and *ranking* hypotheses is a crucial aspect of the MHT algorithm because it is required to differentiate likely and unlikely hypotheses. In the following paragraphs the MHT hypothesis rating equation is derived. It gives the probability of a hypothesis Ω_i^k, given the *cumulative* set of measurements Z^k originating from a *type 1 sensor*:

$$P_i^k := P(\Omega_i^k \big| Z^k)$$

A type 1 sensor is a sensor that is capable of providing information about the number of targets in the area of coverage of the sensor. Such a sensor generates in each scan a set of zero or more measurements. A *primary radar*[3] is a good example for such a sensor. In contrast, a *type 2 sensor* does not provide any information about the number of targets present in the area of coverage of the sensor. A *secondary radar*[4] is an example for a type 2 sensor. Due to its dependence on the transponder, it is impossible to imply the number of targets within the area of coverage of the sensor because the radar would not detect the target unless its transponder is switched on [Mei08].
For a type 1 sensor all measurements of a scan are regarded and processed at the same time, whereas for a type 2 sensor one measurement is processed at a time [Rei79]. The hypothesis rating equation, derived below, is for a type 1 sensor because this sort of sensor is used in person tracking systems.

Given the cumulative set of measurements up to scan k, the probability of a new hypothesis Ω_i^k is composed of the new set of assignments $\omega_i(k)$ and the parent hypothesis $\Omega_{m(i)}^{k-1}$ which contains the assignments of measurements up to and including scan $k-1$. The function $m(i)$ gives the index of the parent hypothesis of a hypothesis Ω_i^k from the

[3] *Primary radar* systems retrieve information about objects within its area of coverage only by detecting radio signals passively reflected by these objects.

[4] *Secondary radar* systems, in contrast to primary radar systems, entirely rely on a *transponder* contained in any object that is supposed to be monitored. These transponders are responsible for sending radio signals that can be detected by secondary radar systems.

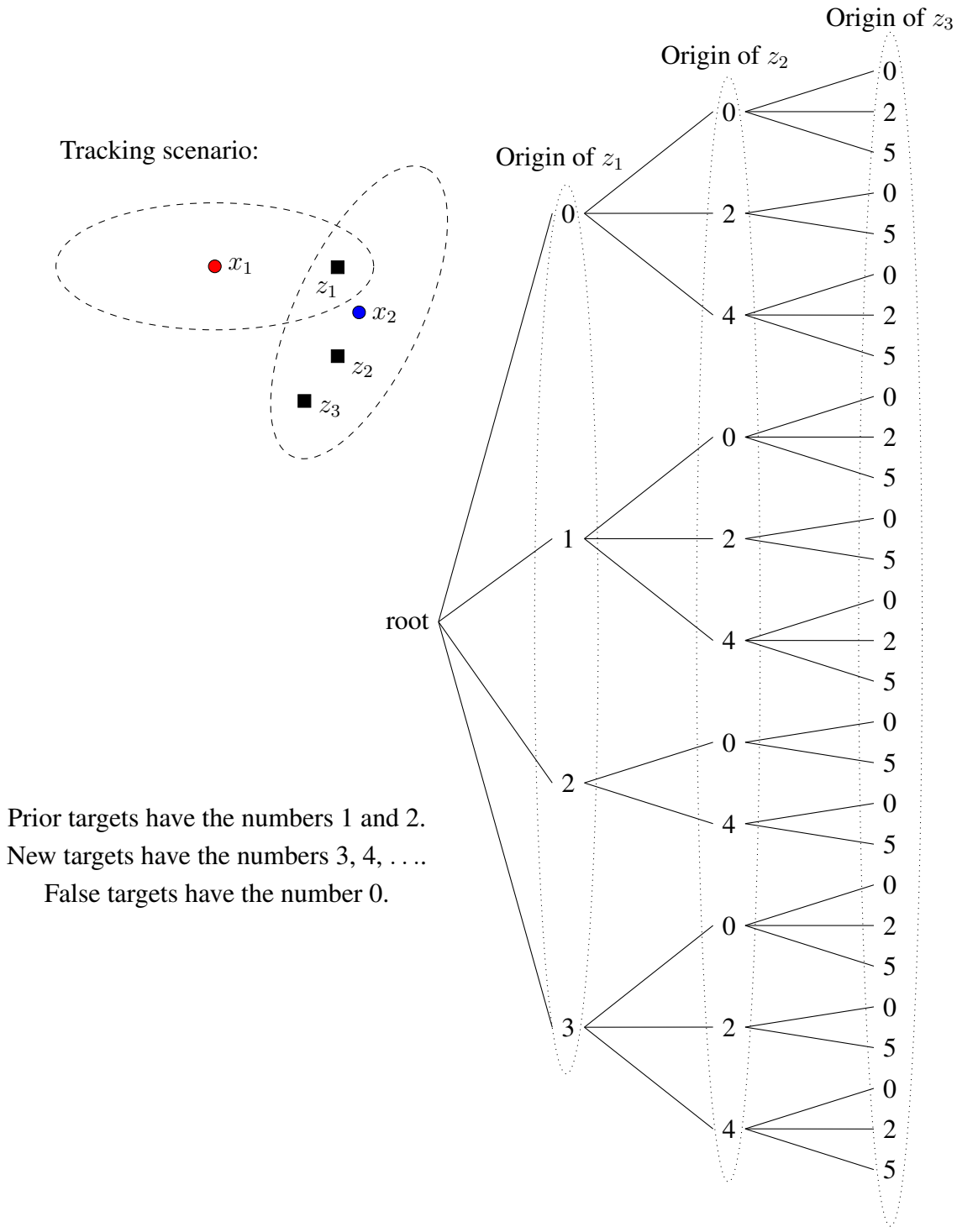


Figure 4.4: An exemplary hypothesis tree for the tracking scenario depicted on the left. Each branch from the root node to one of the leaf nodes represents a different association hypothesis [Rei79].

last scan. Therefore, the new hypothesis is the union of the old and current set of assignments. This is formally written as:

$$\Omega_i^k = \left\{ \Omega_{m(i)}^{k-1}, \omega_i(k) \right\}.$$

The probability P_i^k of a hypothesis Ω_i^k can now be obtained in a recursive manner by using previously known information and new one. By applying first the definition of *conditional probabilities*[5] and second the *chain rule* for conditional probabilities[6], an equation for calculating the probability of a hypothesis $P(\Omega_i^k | Z^k)$ can be derived:

$$\begin{aligned}
P_i^k &= P\left(\Omega_i^k | Z^k\right) \\
&= P\left(\Omega_{m(i)}^{k-1}, \omega_i(k) \middle| Z^{k-1}, Z(k)\right) \\
&= P\left(\Omega_{m(i)}^{k-1}, \omega_i(k), Z^{k-1}, Z(k)\right) \cdot \frac{1}{P(Z^{k-1}, Z(k))} \\
&= \frac{P\left(Z(k) \middle| \Omega_{m(i)}^{k-1}, \omega_i(k), Z^{k-1}\right) P\left(\omega_i(k) \middle| \Omega_{m(i)}^{k-1}, Z^{k-1}\right) P\left(\Omega_{m(i)}^{k-1} \middle| Z^{k-1}\right) P\left(Z^{k-1}\right)}{P(Z^{k-1}, Z(k))} \\
&= \overbrace{\frac{1}{c}}^{\text{evidence}} \underbrace{P\left(Z(k) \middle| \Omega_{m(i)}^{k-1}, \omega_i(k), Z^{k-1}\right)}_{(1)} \overbrace{\underbrace{P\left(\omega_i(k) \middle| \Omega_{m(i)}^{k-1}, Z^{k-1}\right)}_{(2)}}^{\text{likelihood}} \overbrace{\underbrace{P\left(\Omega_{m(i)}^{k-1} \middle| Z^{k-1}\right)}_{(3)}}^{\text{prior}}
\end{aligned} \tag{4.4}$$

By applying the definition of conditional probabilities, the normalizing constant $c = \frac{P(Z^{k-1}, Z(k))}{P(Z^{k-1})} = P(Z^k | Z^{k-1})$ is obtained. It specifies the probability of the occurrence of all measurements up to scan k. The constant never has to be calculated explicitly because it is independent of any hypothesis probability and hence, it is constant among all hypotheses of one iteration.

In order to arrive at a closed equation for the probability P_i^k of a hypothesis Ω_i^k, the three factors of the righthand side of equation (4.4) have to be derived.
Deriving factor (3) of that equation is trivial as it is just the probability of the parent hypothesis $\Omega_{m(i)}^{k-1}$. It is available from the previous iteration. The remaining two terms (1) and (2) describe the likelihood of the new assignment and they are more difficult to evaluate.

[5] The definition of conditional probabilities is $P(x|y) = \frac{P(x,y)}{P(y)}$.

[6] The *chain rule* for conditional probabilities is $P(x_1, \ldots, x_n) = P(x_1 | x_2, \ldots, x_n) \cdot \ldots \cdot P(x_{n-1} | x_n) \cdot P(x_n)$.

Factor (1) of equation (4.4) represents the likelihood of the measurements $Z(k)$ of the current scan, given the new association hypothesis $\Omega_i^k = \{\Omega_{m(i)}^{k-1}, \omega_i(k)\}$ which combines the old and new data associations. It can be obtained as

$$P\left(Z(k)\middle|\Omega_{m(i)}^{k-1}, \omega_i(k), Z^{k-1}\right) = \prod_{i=1}^{M_k} a(z_i(k)), \tag{4.5}$$

where $a : \mathbb{R}^{D_m} \to \mathbb{R}_0^+$ is defined as

$$a(z) = \begin{cases} \frac{1}{V}, & \text{if } z \text{ is a false alarm or a new target} \\ f_{\mathcal{N}\left(Hx_j^k, S_j^k\right)}(z), & \text{if } z \text{ continues a track } j \text{ confirmed by } \Omega_{m(i)}^{k-1}. \end{cases} \tag{4.6}$$

Here, $V \in \mathbb{R}_+$ is the scan volume covered by the sensor, x_j^k is the predicted state of the j^{th} track of hypothesis $\Omega_{m(i)}^{k-1}$ and S_j^k is the corresponding innovation covariance matrix. The *spatial* distribution of false alarms is assumed to be uniform within the area V covered by the sensor. The number of false alarms per time-step is modeled by a Poisson distribution [Rei79].

Factor (2) of equation (4.4) constitutes the probability of the new association hypothesis given its parent hypothesis. Its probability is composed of the probability for the counts N_{DT}, N_{FT} and N_{NT} given $\Omega_{m(i)}^{k-1}$, the probability of a specific *configuration* given the counts N_{DT}, N_{FT} and N_{NT} and lastly the probability of an *assignment* for a given configuration. In that context a configuration is the classification of all M_k measurements of the current scan into measurements assigned to previously known tracks, false alarms or new tracks. An assignment is then a concrete allocation of tracks to those measurements that were assigned to some previously known tracks. Formally, this can be stated as:

$$\begin{aligned} P\left(\omega_i(k)\middle|\Omega_{m(i)}^{k-1}, Z^{k-1}\right) = & P\left(N_{DT}, N_{FT}, N_{NT}\middle|\Omega_{m(i)}^{k-1}\right) \cdot \\ & P(\text{configuration}|N_{DT}, N_{FT}, N_{NT}) \cdot \\ & P(\text{assignment}|\text{configuration}). \end{aligned} \tag{4.7}$$

Now, the three factors of equation (4.7) are derived. It is assumed that the number of previously known tracks that were detected in the current scan N_{DT}, follows a *binomial distribution* $\mathcal{B}(n,p)$. The number of new targets N_{NT} and false targets N_{FT} follow a *Poisson distribution* $\mathcal{P}(\lambda)$. For more details about these distributions refer to section A.1.

With these assumptions the probability of the first factor of equation (4.7) can be written as:

$$P\left(N_{DT},N_{FT},N_{NT}\middle|\Omega_{m(i)}^{k-1}\right) = f_{\mathcal{B}(N_{TGT},P_D)}(N_{DT}) f_{\mathcal{F}(\beta_{FT}V)}(N_{FT}) f_{\mathcal{F}(\beta_{NT}V)}(N_{NT}). \tag{4.8}$$

Now, the probability of the second factor of equation (4.7) is derived. In this one and in a great number of the following derivations the *binomial coefficient*[7] $\binom{n}{k}$ is used to count the number of possible combinations. The total number of different ways to assign N_{DT} out of M_k measurements to prior targets, N_{FT} out of M_k measurements to false targets and N_{NT} out of M_k measurements to new targets is given by the following expression:

$$\underbrace{\binom{M_k}{N_{DT}}}_{\text{to prior targets}} \cdot \underbrace{\binom{M_k - N_{DT}}{N_{FT}}}_{\text{to false targets}} \cdot \underbrace{\binom{M_k - N_{DT} - N_{FT}}{N_{NT}}}_{\text{to new targets}}.$$

Hence, the probability of a configuration given the counts N_{DT}, N_{FT} and N_{NT} is:

$$P(\text{configuration}|N_{DT},N_{FT},N_{NT}) = \frac{1}{\binom{M_k}{N_{DT}}\binom{M_k-N_{DT}}{N_{FT}}\binom{M_k-N_{DT}-N_{FT}}{N_{NT}}}. \tag{4.9}$$

Finally, the last factor of equation (4.7), namely the number of possible combinations for assigning the N_{DT} measurements to the N_{TGT} prior targets has to be derived. N_{FT} and N_{NT} do not have to be considered because the number of possible assignments for both equals 1.

$$\frac{N_{TGT}!}{(N_{TGT} - N_{DT})!}$$

Therefore, the probability for an assignment given a certain configuration is:

$$P(\text{assignment}|\text{configuration}) = \frac{1}{\frac{N_{TGT}!}{(N_{TGT}-N_{DT})!}} = \frac{(N_{TGT} - N_{DT})!}{N_{TGT}!}. \tag{4.10}$$

By substituting equations (4.8), (4.9) and (4.10) into equation (4.7) and simplifying the resulting expression, the final equation for the probability of the new association hypothesis Ω_i^k is obtained.

$$\begin{aligned} &P\left(\omega_i(k)\middle|\Omega_{m(i)}^{k-1},Z^{k-1}\right) \\ =&f_{\mathcal{B}(N_{TGT},P_D)}(N_{DT}) f_{\mathcal{F}(\beta_{FT}V)}(N_{FT}) f_{\mathcal{F}(\beta_{NT}V)}(N_{NT}) \cdot \end{aligned}$$

[7] The binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ answers an important question of combinatorics. It gives the number of ways k items can be chosen out of n items, disregarding their order. To put it more formally, the binomial coefficient gives the number of k-element subsets of a set with n elements.

$$
\begin{aligned}
&\frac{1}{\binom{M_k}{N_{DT}}\binom{M_k-N_{DT}}{N_{FT}}\binom{M_k-N_{DT}-N_{FT}}{N_{NT}}} \cdot \frac{(N_{TGT}-N_{DT})!}{N_{TGT}!} \\
=&\binom{N_{TGT}}{N_{DT}} {P_D}^{N_{DT}}(1-P_D)^{N_{TGT}-N_{DT}} f_{\mathcal{F}(\beta_{FT}V)}(N_{FT}) f_{\mathcal{F}(\beta_{NT}V)}(N_{NT}) \cdot \\
&\frac{1}{\frac{M_k!}{N_{DT}!\cancel{(M_k-N_{DT})!}}\frac{\cancel{(M_k-N_{DT})!}}{N_{FT}!\cancel{(M_k-N_{TD}-N_{FT})!}}\frac{\cancel{(M_k-N_{DT}-N_{FT})!}}{N_{NT}!(M_k-N_{DT}-N_{FT}-N_{NT})!}} \cdot \frac{(N_{TGT}-N_{DT})!}{N_{TGT}!} \\
=&\frac{\cancel{N_{TGT}!}}{\cancel{N_{DT}!}\cancel{(N_{TGT}-N_{DT})!}} {P_D}^{N_{DT}}(1-P_D)^{N_{TGT}-N_{DT}} f_{\mathcal{F}(\beta_{FT}V)}(N_{FT}) f_{\mathcal{F}(\beta_{NT}V)}(N_{NT}) \cdot \\
&\frac{\cancel{N_{DT}!}}{M_k!} N_{FT}! N_{NT}! \underbrace{(M_k-N_{DT}-N_{FT}-N_{NT})!}_{=1} \frac{\cancel{(N_{TGT}-N_{DT})!}}{\cancel{N_{TGT}!}} \\
=&\frac{N_{FT}! N_{NT}!}{M_k!} {P_D}^{N_{DT}}(1-P_D)^{N_{TGT}-N_{DT}} f_{\mathcal{F}(\beta_{FT}V)}(N_{FT}) f_{\mathcal{F}(\beta_{NT}V)}(N_{NT}).
\end{aligned}
$$

Substituting this result together with equation (4.5) into equation (4.4) the formula for determining the probability of a hypothesis can be finally obtained:

$$
\begin{aligned}
P_i^k =&\frac{1}{c}\frac{N_{FT}! N_{NT}!}{M_k!} {P_D}^{N_{DT}}(1-P_D)^{N_{TGT}-N_{DT}} f_{\mathcal{F}(\beta_{FT}V)}(N_{FT}) f_{\mathcal{F}(\beta_{NT}V)}(N_{NT}) \\
&\cdot \left(\prod_{i=1}^{N_{DT}} f_{\mathcal{N}\left(Hx_j^k, S_j^k\right)}(z_i(k))\right) \frac{1}{V^{N_{FT}+N_{NT}}} P_{m(i)}^{k-1} \\
=&\frac{1}{c}\frac{\cancel{N_{FT}!}\cancel{N_{NT}!}}{M_k!} {P_D}^{N_{DT}}(1-P_D)^{N_{TGT}-N_{DT}} \frac{(\beta_{FT}\cancel{V})^{\cancel{N_{FT}}}}{N_{FT}!} e^{-\beta_{FT}V} \frac{(\beta_{NT}\cancel{V})^{\cancel{N_{NT}}}}{N_{NT}!} e^{-\beta_{NT}V} \\
&\cdot \left(\prod_{i=1}^{N_{DT}} f_{\mathcal{N}\left(Hx_j^k, S_j^k\right)}(z_i(k))\right) \frac{1}{\cancel{V^{N_{FT}+N_{NT}}}} P_{m(i)}^{k-1} \\
=&\underbrace{\frac{1}{c}\frac{e^{-\beta_{FT}V} e^{-\beta_{NT}V}}{M_k!}}_{=:\frac{1}{c'}} {P_D}^{N_{DT}}(1-P_D)^{N_{TGT}-N_{DT}} \beta_{FT}^{N_{FT}} \beta_{NT}^{N_{NT}} \\
&\cdot \left(\prod_{i=1}^{N_{DT}} f_{\mathcal{N}\left(Hx_j^k, S_j^k\right)}(z_i(k))\right) P_{m(i)}^{k-1} \\
=&\frac{1}{c'} {P_D}^{N_{DT}}(1-P_D)^{N_{TGT}-N_{DT}} \beta_{FT}^{N_{FT}} \beta_{NT}^{N_{NT}} \left(\prod_{i=1}^{N_{DT}} f_{\mathcal{N}\left(Hx_j^k, S_j^k\right)}(z_i(k))\right) P_{m(i)}^{k-1}. \quad (4.11)
\end{aligned}
$$

The dependence on V got eliminated by the substitution by the Poisson distribution. The constant c was condensed into the constant $c' = cM_k!/(e^{-\beta_{FT}V} e^{-\beta_{NT}V})$.

Equation (4.11) is the mathematical key result of Reid's paper. Thanks to its dependence on the probability of the previous time-step, the equation can be applied iteratively

to calculate the probability of each data association hypothesis.

Although the formula (4.11) is complex its application in practice is not too hard. The required steps are to multiply first all prior hypothesis probabilities by $(1 - P_D)^{N_{TGT}}$. Then, as a branch is created for each measurement and its hypothesized origin, the likelihood of that branch is determined by either multiplying the prior probability by β_{FT} in case of a false alarm, β_{NT} in case of a new target or $P_D f_{\mathcal{N}(Hx_j^k, S_j^k)}(z_i(k))/(1 - P_D)$ in case of a track continuation.

4.3.4 Practical Issues

MHT constitutes a difficult algorithmic problem. The major issue of the MHT algorithm is that its memory and processing requirements are ever-expanding as more measurements are processed because more and more hypotheses have to be maintained. As showed in previous examples, the MHT algorithm is of *exponential* character. Natural limitations in memory and processing power make any direct implementation of the algorithm impossible. In the following strategies which overcome practical issues of the MHT algorithm are devised.

4.3.4.1 Hypothesis Pruning

In general, the number of generated and maintained hypotheses has to be kept in check in order to not exceed processing power and memory. This can be achieved by keeping only some hypotheses and *pruning* the rest. In principle, pruning means discarding certain hypotheses on the basis of some criteria. Pruning is a simple but yet effective optimization. In this section a few pruning strategies are presented.

Count-based Pruning An often applied pruning strategy is to keep only the n-best (most likely) hypotheses and to discard the rest. For doing so after each update the set of hypotheses Ω^k is sorted by hypothesis probability P_i^k and then all hypotheses Ω_i^k with $i \leq n$ are deleted.

The advantage of that strategy is its simplicity. Its disadvantage is that eventually hypotheses are pruned because they are not within the n-best hypotheses, even though their probability is high and it might be worth considering them in follow-

ing scans when more information is available. This particular drawback is solved by *probability-based* pruning.

Probability-based Pruning In probability-based pruning only hypotheses $\Omega_i^k \in \Omega^k$ with a probability P_i^k bigger than a certain threshold $P_{\text{min}} \in [0,1)$ are kept. Thus, all hypotheses with $P_i^k < P_{\text{min}}$ are discarded and all remaining hypotheses are kept.

The advantage of this pruning strategy is that probable hypotheses are not discarded. Its disadvantage is that the number of hypotheses can vary considerably which affects the time required to process a scan.

n-scan-back Pruning This pruning strategy is based on the assumption that ambiguities during the data association process can only be resolved within the next n time-steps. Therefore, at time-step $k+n$ a final association decision has to be made and just one node is kept. All the other child nodes of the node at time-step k are discarded. The pruning decision is made on the basis of the probability obtained by summing up the hypothesis probabilities of all leaves associated with a node. The tree branch containing the leaf nodes with the maximum probability is kept because it indicates the most probable branch. In figure 4.5 an example of n-scan-back pruning with $n = 2$ is given [Can08].

Hypothesis Merging It might happen that several hypotheses with different histories result in very similar track estimates. Such hypotheses should be merged because they represent almost the same measurement hypothesizing but take up space for other likely hypotheses.

Two hypotheses can be merged if they both contain the same number of tracks and the contained tracks are similar. The similarity of two tracks can be determined e. g. on the basis of the distance of their state predictions $d_{sp} \in \mathbb{R}_0^+$ and on the distances of the eigen values $d_{\lambda_i} \in \mathbb{R}_0^+$ of their associated covariance matrices. If these distances are smaller than the thresholds ϵ_{sp}, $\epsilon_\lambda \in \mathbb{R}_0^+$, both hypotheses are merged into a single one. The probability of the new hypothesis is set to the sum of the probabilities of the merged hypotheses. The state prediction and covariance matrix of the new hypothesis are set to the average of the state predictions and covariance matrices of the merged hypotheses.

In order to merge a set of many similar hypotheses two of them are iteratively

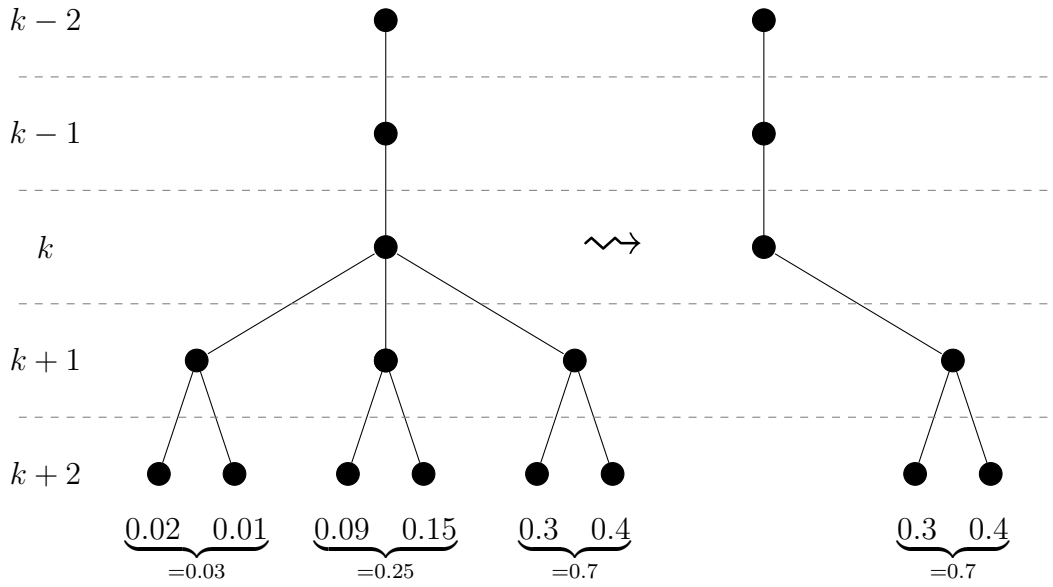


Figure 4.5: An example of the n-scan-back pruning strategy. The lefthand side shows a hypothesis tree prior to pruning. The righthand side shows the same hypothesis tree after n-scan-back pruning was applied [Can08].

chosen, removed from the set and merged into a new hypothesis which is put back into the set. This procedure is repeated until only one hypothesis is left in the set.

4.3.4.2 Clustering

Spatially separated tracks that share no common measurements in their associated gates for some time-steps or even for their whole life time often emerge when tracking multiple targets. As tracking performance mainly depends on the number of tracks present within each hypothesis, a division of these tracks into multiple independent local hypotheses, so called *clusters*, that do not interact with each other can be of a great value. This allows solving a number of small tracking problems instead of a big one, which reduces the amount of required memory and computation time significantly. Moreover, this partitioning scheme is especially nowadays beneficial because all computations on individual clusters can be performed in parallel without any synchronization primitives thanks to the independence of the clusters. This allows the efficient utilization of modern *multi-core* CPUs.

Tracks of the same cluster share common measurements, whereas tracks of different

cluster do not. A cluster is completely defined by the set of tracks and measurements it contains, as well as by the set of hypotheses. The three operations required for cluster management are outlined in the following [dW01, Kur90, Bla04, Rei79]:

Cluster Formation A new measurement is associated with a cluster if it falls within the gating region of any track contained in that cluster. If a measurement cannot be associated with any existing cluster, a new cluster containing this measurement is formed.

Cluster Combining If a new measurement gates with tracks of two or more different clusters, these clusters are combined into one *super cluster*. The set of hypotheses of the super cluster is formed by building every possible joint hypothesis of the hypotheses of the clusters to be combined. The joint hypothesis probabilities are calculated by multiplying the probabilities of the combined hypotheses. This process is illustrated in figure 4.6. In practice the hypothesis set of the super cluster has to be pruned. In fact, it can be iteratively pruned after each combination of two clusters. By doing so the total number of hypotheses that have to be generated for the super cluster is significantly reduced.

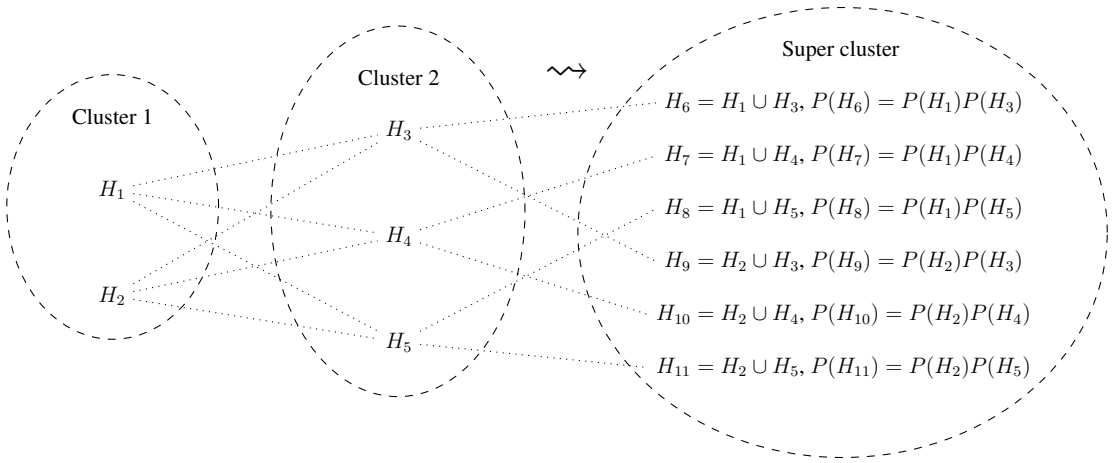


Figure 4.6: An example for the combination of two previously independent clusters.

Cluster Splitting Clusters can be reduced in the number of targets through the process of cluster splitting. A track can be removed from a cluster and put into its own cluster, if there is a measurement in a scan which gates only with that particular track and with no other track of any other cluster.

4.3.4.3 Direct Hypothesis Generation

Although all previously mentioned optimizations are employed, there remains the inherent exponential phase of hypothesis generation before any hypothesis reduction technique can be applied. In order to overcome this problem the conceptional MHT algorithm is reformulated so that the best hypotheses can be directly determined instead of exhaustively enumerating all of them. This optimization is presented in the next chapter.

4.4 n-best Multiple Hypothesis Tracking

4.4.1 Introduction

As mentioned in section 4.3.4.3 Reid's MHT algorithm is inherently exponential in the number of hypotheses which have to be identified and evaluated, even if all previously presented optimization techniques are applied. The reason for this is that during the generation of hypotheses, one with a higher probability than any previously enumerated might appear at any point in time. Today's computers are not powerful enough to exhaustively identify all possible hypotheses, especially in high spatial density and high clutter environments. This makes the original MHT formulation of Reid not applicable on most practical tracking scenarios.
To overcome this problem the direct computation of only the n-best hypotheses is performed nowadays, instead of identifying all possible hypotheses. Danchick and Newnam (2006) describe how to exploit *Murty's algorithm* from the field of operations research to directly solve for the n-best hypotheses [DN06]. Originally this idea dates back to the work of Cox and Miller (1995) [CM95].

Directly solving for the n-best hypotheses is all but trivial. The problem is reduced to a *linear assignment problem* (LAP) from the field of mathematical optimization. In order to understand the n-best MHT approach it is necessary to introduce the LAP and the *Hungarian method* as one algorithm for determining a solution. After that, Murty's algorithm to solve for an arbitrary number of ranked best solutions is presented. Finally, it is shown how the conceptional MHT algorithm can be reformulated to directly solve for the n-best hypotheses using Murty's algorithm.

4.4.2 The Linear Assignment Problem

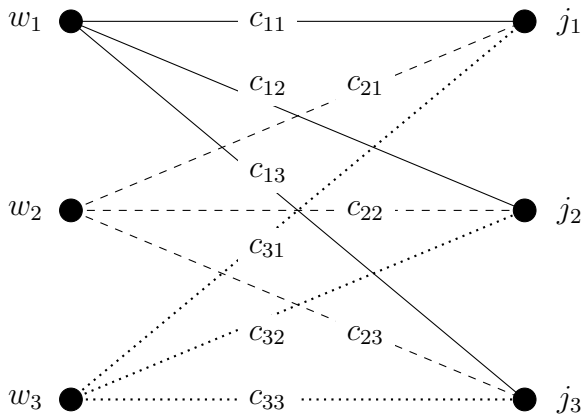
To formally state the *linear assignment problem* (LAP) and to present an algorithm for solving it, a few important definitions from the field of *graph theory* are required. For an overview refer to section A.2.

4.4.2.1 Problem Definition

The classical LAP deals with the problem of how to assign $n \in \mathbb{N}$ *workers* to n *jobs* in the best possible manner (e. g. the cheapest or the most efficient). An assignment problem is completely specified by two components: the assignments representing the combinatorial structure and a *cost function* giving the cost of an assignment. In the traditional formulation the LAP is *squared* which means the number of workers is equal to the number of jobs. In that case the number of feasible solutions is $n!$ [Bc99, BDM09].

The LAP can be formulated as a *weighted bipartite matching problem*. Given a bipartite graph $G = (V = X \cup Y,E)$ with an associated cost function $w : E \to \mathbb{R}$, a perfect matching $M^\star$ with minimum cost is to be found. That means that for all other matchings $M' \neq M^\star$ of G it holds that $w(M^\star) \leq w(M')$. Then $M^\star$ is called a *min-weight* matching. The set of vertices X and Y describe the workers and jobs respectively and the matching $M^\star$ describes the assignment. In the following sections the terms matching and assignment are used interchangeably. Without a loss of generality it can be assumed that the graph G is complete. If G is not complete, edges with cost $\max_{e \in E} \{w(e)\} + 1$ can be added without changing the resulting matching.

The LAP cannot only be described in terms of bipartite graphs. Another often seen representation is a cost matrix $C \in \mathbb{R}^{n \times n}$, where each $c_{i(x),i(y)}$ is the cost for assigning vertex $x \in X$ to vertex $y \in Y$. An assignment can be represented either by a second matrix $A \in \{0,1\}^{n \times n}$, where $a_{i(x),i(y)} = 1$ if $x \in X$ was assigned to $y \in Y$ and 0 otherwise, or by a set of matrix element indices $\alpha = \{(i_1,j_1),\ldots,(i_n,j_n)\}$, in which no two distinct members have the same row or column index: $(r,s),(t,u) \in \alpha \Rightarrow r \neq t$ and $s \neq u$ [Bc98, DN06]. The index function $i : V \to \{1,\ldots,n\}$ maps vertices to matrix cells. In figure 4.7 the two representations are depicted.



$$C = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

Figure 4.7: A LAP of size 3 represented as a bipartite graph (left) and a cost matrix (right).

There exist many polynomial time algorithms of different complexity and speed to solve the LAP. As a discussion of all of them would go beyond the scope of this thesis, only the *Hungarian method* as the most well-known LAP solving algorithm is discussed as a representative in the next section.

4.4.2.2 The Hungarian Method

The *Hungarian method* (also known as *Kuhn-Munkres algorithm*) is an algorithm for solving weighted, linear assignment problems. It was first published by Kuhn (1955) [Kuh55] who used ideas of the Hungarian mathematicians Dénes Kőnig and Jenő Egerváry. Munkres (1957) refined the algorithm and provided a complexity analysis [Mun57].

The Hungarian method described in the rest of this section calculates a max-weight matching. However, it is easy to change the assignment costs slightly in order to obtain a min-weight matching. All that has to be done is to apply the following transformation:

$$w(x,y) \leftarrow \max_{(a,b)\in E} w(a,b) - w(x,y), \forall (x,y) \in E. \tag{4.12}$$

In order to understand the Hungarian method an important theorem is derived first, on which the rest of the algorithm is based. It provides the connection between *equality graphs* and max-weight matchings. This theorem is important because it transforms the *optimization* problem of finding a max-weight matching into the *combinatorial* problem of finding a *perfect matching*. This is a classical technique applied to solve optimization problems [Jun07].

Theorem 1. *If the equality graph G_l of a graph $G = (V,E)$ has a perfect matching $M^\star \subseteq G_l$, then $M^\star$ is a max-weight matching.*

Proof. It has to be shown that no other *complete matching* $M' \subseteq G$ can have a weight smaller than $M^\star$.

1. By definition, in a perfect matching each vertex is covered exactly once. Hence, $w(M^\star) = \sum_{e \in M^\star} w(e) = \sum_{v \in V} l(v)$, where $l(v)$ denotes the vertex labeling of the equality sub-graph G_l.
2. Any other matching M' in G satisfies $w(M') = \sum_{e \in M'} w(e) \leq \sum_{v \in V} l(v)$.
3. Thus, $w(M') \leq w(M^\star) \Rightarrow M^\star$ must be a max-weight matching.

□

The Hungarian method starts with an arbitrary but feasible labeling l and an arbitrary matching M of G_l. While M is not a perfect matching its size is increased with an *augmenting path*. If no augmenting path exists, l is improved to l' so that $G_l \subset G_{l'}$ [Jun07]. The details of all undertaken steps are given below:

1. Initialize a vertex labeling l according to equation (A.7) and determine the corresponding equality sub-graph G_l.
2. Pick an arbitrary matching M in G_l.
3. If M is a perfect matching (theorem 1), also a max-weight matching was found and the algorithm terminates (goto step 6). Otherwise, some unmatched vertex $x \in X$ exists still. Set $S = \{x\}$ and $T = \emptyset$.
4. If $N_l(S) = T$, find $\alpha_l = \min_{x \in S,\, y \notin T}\{l(x) + l(y) - w(x,y)\}$ and update the vertex labeling in the following manner:

$$l'(v) \leftarrow \begin{cases} l(v) - \alpha_l, & v \in S \\ l(v) + \alpha_l, & v \in T \\ l(v), & \text{otherwise} \end{cases}$$

5. If $N_l(S) \neq T$, pick $y \in N_l(S) \setminus T$ and perform the following case differentiation:

- If y is *free*, the path between u and y is augmenting. Augment M and go to step 2.
- If y is *matched* to z, update $S \leftarrow S \cup \{z\}$ and $T \leftarrow T \cup \{y\}$ and go to step 3.

6. Done.

The algorithm always terminates because in each step either the size of M or G_l is increased. When the algorithm terminates, M will be a perfect matching as well as a max-weight matching because of theorem 1. The Hungarian method solves the LAP in $\mathcal{O}(|V|^3)$ [BDM09].

4.4.2.3 The Rectangular Linear Assignment Problem

Assignment problems where the number of workers m is different from the number of jobs n (but $m \leq n$) are called *rectangular linear assignment problems* (RLAPs). They constitute a generalization of the square case. For rectangular LAPs there exist $\prod_{i=0}^{m-1}(n-i)$ solutions.

A widely used approach to solve a RLAP is to extend it to a regular square LAP by adding dummy vertices and edges to the graph. Nevertheless, this approach is strongly unrecommended because the increased size of the LAP increases the associated computational costs required for solving it as well [BV10].
It is recommended to use algorithms particularly designed for rectangular LAPs. Today, the best algorithm to solve the RLAP is the extended JVC algorithm, called JVR. The original JVC algorithm (1987) was developed by Jonker, Volgenant and Castanon and has a complexity of $\mathcal{O}(m^3)$ [JV87]. It was extended to the rectangular case by Volgenant (1996) [Vol96]. A sufficient presentation of this algorithm would go beyond the scope of this thesis and is hence omitted.

4.4.2.4 Murty's Algorithm

Often not only the very best solution of a LAP is of interest but also the second, third or generally n^{th} best solution for $n = 1,2,\ldots$. Murty (1968) invented an algorithm for determining them [Mur68]. It requires a classical method for solving LAPs as e. g. the

Hungarian method (see section 4.4.2.2). Murty's algorithm guarantees to find the n-best solutions in an efficient, polynomially limited manner. Its complexity is derived in the end of this section.

Murty's algorithm determines a ranked set of the n-best assignments of a given LAP described by a cost matrix $C \in {\mathbb{R}_+}^{m \times m}$, by repeatedly applying a LAP solver to an adapted cost matrix C_N. For doing so, a set[8] of *nodes*[9] U is maintained which facilitates adapting the initial cost matrix C, needed to solve for the i^{th} best solution, $1 \leq i \leq n$. A worker-to-job assignment is called a *cell*. A node N in that context is a 4-tuple which consists of the best assignment $a^\star_N$ for node N, its assignment costs $c(a^\star_N)$, as well as a set of r *fixed* cells F_N and a set of s *excluded* cells E_N:

$$N := (a^\star_N,\, c(a^\star_N),\, \underbrace{\{(g_1,h_1),\ldots,(g_r,h_r)\}}_{\text{fixed assignments } F_N},\, \underbrace{\{(k_1,l_1),\ldots,(k_s,l_s)\}}_{\text{excluded assignments } E_N}) \tag{4.13}$$

Let $\mathcal{A}$ denote the set of all possible assignments. Then N describes a *restricted* set of assignments $\mathcal{A}_N \subseteq \mathcal{A}$, where certain cells are included and others are excluded. This is written as

$$\begin{aligned} \mathcal{A}_N &:= \{a \in \mathcal{A} \,|\, (g_1,h_1),\ldots,(g_r,h_r) \in a \wedge (k_1,l_1),\ldots,(k_s,l_s) \notin a\} \\ &= \{(g_1,h_i),\ldots,(g_r,h_r),\overline{(k_1,l_1)},\ldots,\overline{(k_s,l_s)}\}, \end{aligned} \tag{4.14}$$

where a bar above a cell indicates that it is excluded from the node. Hence, the node N describes all possible assignments that *contain* the cells $(g_1,h_1),\ldots,(i_r,j_r)$ and that do not contain the cells $(k_1,l_1),\ldots,(k_s,l_s)$.

In the first step of Murty's algorithm, also referred to as *stage 1*, the very best assignment $a^\star_{N_1} = \{(p_1,q_1),\ldots,(p_n,q_n)\}$ with cost $c(a^\star_{N_1})$ of C is determined with the use of a classical LAP solver. Afterwards, it is used to initialize the set U:

$$U = \{(a^\star_{N_1},\, c(a^\star_{N_1}),\, \emptyset,\, \emptyset)\}. \tag{4.15}$$

In the second part of the algorithm, also referred to as *stage 2*, the node in U with least costs is repeatedly determined, removed from U, inserted into the result list R and finally *partitioned*. These operations are performed until U is empty or the n-best solutions are

[8] In practice this would be a priority queue.

[9] In the rest of this chapter the term *node* refers to a somewhat different mathematical object than in the original paper of Murty. This decision was made in order to come up with a more practical, algorithmic description of the algorithm.

found and thus $|R| = n$.
Partitioning a node N by an assignment a involves the creation of $r-1$ new nodes N_i, where the restricted set of assignments $\mathcal{A}_{N_i}$ of each child node does not contain the assignment a and it holds that $\cup_{i=1}^{r-1}\mathcal{A}_{N_i} = \mathcal{A} \setminus \{a\}$. Partitioning a node N by an assignment $a = \{(g_1,h_1),\ldots,(g_r,h_r)\}$ is done in the following way:

$$N_i = \left(a^\star_{N_i},\, c(a^\star_{N_i}),\, F_N \cup \{(g_i,h_i),\ldots,(g_{r-1},h_{r-1})\},\, E_N \cup \{(g_i,h_i\}\right),\ i = 1,\ldots,r-1. \tag{4.16}$$

It should be mentioned that only $r-1$, not r, new nodes have to be generated because the cost matrix of the last partition N_r would only be of size 1×1 with $(C_{N_r})_{11} = \infty$. Such a matrix does not contain any feasible assignment and hence only $r-1$ child nodes are generated.

In order to construct the remaining cost matrix $C_N \in {\mathbb{R}_+}^{(n-r)\times(n-r)}$ of a node N the following changes need to be applied on the initial cost matrix C. First, all rows $g_1,\ldots,g_r$ and all columns $h_1,\ldots,h_r$ have to be striked off to fix these cells in all successive assignments of the node. Second, all matrix cells $(k_1,l_1),\ldots,(k_s,l_s)$ have to be replaced by ∞ to exclude these cells in all successive assignments of the node. The whole algorithm is shown in pseudo code in algorithm 4.2.

Algorithm 4.2 Murty's algorithm

Input: $C \in {\mathbb{R}_+}^{m\times m}$ and $n \in \mathbb{N}$
Output: Set R of n-best assignments

1: Determine best assignment $a^\star_{N_1}$ with cost $c(a^\star_{N_1})$ of C
2: Initialize set $U = \{(a^\star_{N_1},\, c(a^\star_{N_1}),\, \emptyset,\, \emptyset)$
3: **while** $U \neq \emptyset$ **or** $|R| \leq n$ **do**
4: $\quad N_{\text{min}} \leftarrow$ Node in U with minimum cost
5: $\quad U \leftarrow U \setminus \{N_{\text{min}}\}$
6: $\quad R \leftarrow R \cup \{N_{\text{min}}\}$
7: $\quad \{N_1,\ldots,N_{r-1}\} \leftarrow$PARTITION($N_{\text{min}}$)
8: $\quad U \leftarrow U \cup \{N_1,\ldots,N_{r-1}\}$
9: **end while**
10: **return** R

Even though not explicitly mentioned in the original work of Murty, the algorithm is

as well applicable with minor changes to rectangular assignment problems [CM95]. For rectangular matrices r, instead of $r - 1$, partitions have to be generated because the last partition might still contain valid assignments.

Lastly, the algorithmic worst-case complexity of Murty's algorithm in conjunction with the JVC solver is derived [PPBS99, MS$^+$97].
When solving for the n-best assignments each assignment creates in the worst-case m new sub-problems (nodes). Hence, up to $\mathcal{O}(nm)$ nodes have to be inserted into the priority queue U and up to $\mathcal{O}(nm)$ LAPs have to be solved. Solving a single LAP has a worst-case complexity of $\mathcal{O}(m^3)$. Inserting a single node into U has a worst-case complexity of $\mathcal{O}(nm)$. As such, the worst-case complexity of Murty's algorithm presents as follows:

$$\mathcal{O}(nm(m^3 + nm)) = \mathcal{O}(nm^4). \tag{4.17}$$

4.4.2.4.1 Example In figure 4.8 an example of Murty's algorithm for a squared LAP of size 3 is depicted. All five feasible assignments are calculated in a ranked fashion. Solid nodes are inserted into U. Dashed nodes have a cost of ∞ and are thus discarded. The hierarchical character of the algorithm is illustrated by the tree of generated nodes. In each partitioning step a new set of child nodes is generated for the partitioned node.

4.4.3 Reformulated Multiple Hypothesis Tracking

After the introduction of the LAP and Murty's algorithm the n-best MHT approach can be presented. Given the set of all hypotheses Ω^k at time-step k and a set of new measurements $Z(k + 1)$, the aim is to directly compute the n-best posterior hypotheses without enumerating all possible posterior hypotheses exhaustively.

The task can be accomplished by constructing for all n prior hypotheses $\Omega_i^k \in \Omega^k$ a *data association matrix* L and determining the n-best posterior hypotheses for each Ω_i^k with Murty's algorithm. After obtaining the n^2-best hypotheses for all n prior hypotheses only the n-best are kept and the remaining $n^2 - n$ hypotheses are discarded. Instead of n, n^2 hypotheses have to be generated because no hypothesis history is encoded in the data association matrices. The data association matrices only account for the association

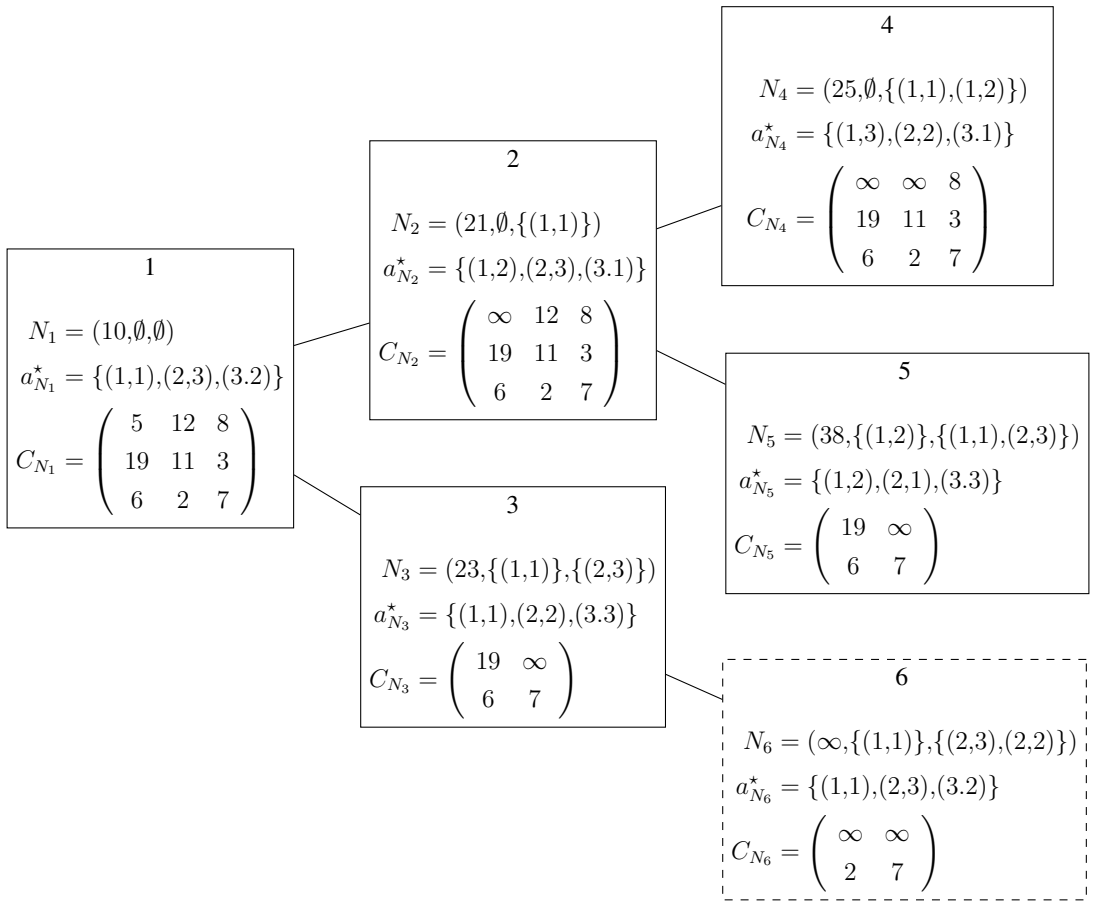


Figure 4.8: An example of applying Murty's algorithm on a 3×3 matrix. Solid nodes are inserted into U. Dashed nodes have a cost of ∞ and are thus discarded.

probabilities of the current scan and they do not incorporate probabilities of prior hypotheses. In figure 4.9 the basic structure of the n-best MHT approach is illustrated. Consequently, to recast Reid's conceptional MHT algorithm into a LAP two things must be obtained:

1. The hypothesis generation procedure has to be reformulated so that it is possible to apply Murty's algorithm to solve for the n-best posterior hypotheses.

2. A formula is required which gives the number of all feasible posterior hypotheses because there might exist more feasible assignments than feasible posterior hypotheses for a given hypothesis and for a given set of measurements. In that case the number of solutions that Murty's algorithm generates has to be limited to the maximum number of existing posterior hypotheses.

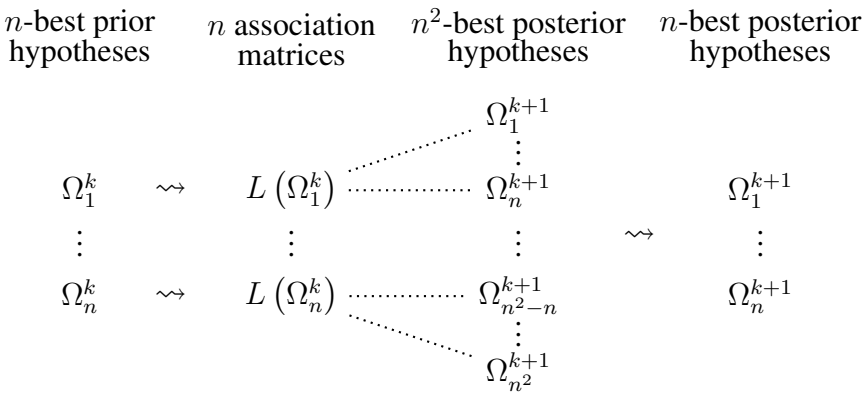


Figure 4.9: Structure of the n-best MHT approach.

In the next sections first a formula for counting the number of posterior hypotheses is derived, followed by the presentation of the data association matrix casting the conceptional MHT into a LAP. Finally, the algorithm illustrated in figure 4.9 is given as pseudo-code.

4.4.3.1 Counting Posterior Hypotheses

Given a prior hypothesis Ω_i^{k-1} and $|Z(k)| = M_k$ new measurements, the number of feasible posterior hypotheses N_i^k can be obtained by accounting for all measurement-to-target association possibilities and by considering the configuration of the prior hypothesis.

(i) $0 \leq N_{DT} \leq \min(M_k, N_{TGT})$ measurements can be associated with existing tracks. For each possible number of associations N_{DT} there are $\binom{N_{TGT}}{N_{DT}}$ ways to choose N_{DT} existing tracks.

(ii) The first of the N_{DT} tracks can then be assigned to M_k different measurements, the second track to $M_k - 1$ measurements, etc., till all N_{DT} tracks are assigned. Thus, there are $M_k \cdot (M_k - 1) \cdot \ldots \cdot (M_k - N_{DT} + 1) = \binom{M_k}{N_{DT}} N_{DT}!$ ways of assigning N_{DT} existing tracks to M_k measurements.

(iii) There are still $M_k - N_{DT}$ remaining measurements to be assigned to either a false alarm, or a new target. Thus, there are $2^{M_k - N_{DT}}$ possible assignments left.

Combining the counts of possible assignments (i) – (iii), the following formula is obtained to determine the total number of parent hypotheses N_i^k, given the prior hypothesis

Ω_i^k:

$$N_i^k = \sum_{N_{DT}=0}^{\min(M_k, N_{TGT})} \left[\binom{N_{TGT}}{N_{DT}} \binom{M_k}{N_{DT}} N_{DT}! 2^{M_k - N_{DT}} \right] \tag{4.18}$$

With this formula it is possible only now to get an overview of the combinatorial explosion occurring during Reid's hypothesis generation process. In table 4.2 a few examples are listed for the number of posterior hypotheses to be generated, given a hypotheses with N_{TGT} tracks and a scan containing M_k measurements [DN06].

M_k \ N_{TGT}	1	2	3	4	5	6
1	3	4	5	6	7	8
2	8	14	22	32	44	58
3	20	44	86	152	248	380
4	48	128	304	648	1 256	2 248
5	112	352	992	2 512	5 752	12 032
6	256	928	3 040	8 992	24 064	58 576

Table 4.2: The number of feasible posterior hypotheses for different numbers of existing tracks and new measurements. The combinatorial explosion is clearly visible. Already for 6 tracks and 6 measurements 58 576 posterior hypotheses have to be generated.

4.4.3.2 Data Association Matrix

Next, it is described how the data association matrix L is constructed. It encodes all feasible, posterior data association hypotheses of the original tree-based formulation of Reid for a single hypothesis.

Given a hypothesis Ω_i^k and a set of measurements $Z(k+1)$ of the current scan, the corresponding data association matrix $L\left(\Omega_i^k\right) \in \mathbb{R}^{M_k \times (N_{TGT} + 2M_k)}$ is defined as

$$L\left(\Omega_i^k\right) = (l_{ij}) := \left(\begin{array}{ccc|c|c} l_{11} & \cdots & l_{1,N_{TGT}} & \left(\frac{\beta_{NT}(1-P_D)}{P_D}\right) I & \left(\frac{\beta_{FT}(1-P_D)}{P_D}\right) I \\ \vdots & \ddots & \vdots & & \\ l_{M_k 1} & \cdots & l_{M_k N_{TGT}} & & \end{array} \right), \tag{4.19}$$

where the likelihood for assigning a new measurement $z_i(k+1)$, $1 \leq i \leq M_k$ to an existing track $1 \leq j \leq N_{TGT}$ is given by

$$(l_{ij}) = f_{\mathcal{N}\left(Hx_j^k, S_j^k\right)}(z_i(k)). \tag{4.20}$$

This cost matrix accounts for measurement-to-track, measurement-to-new-target and measurement-to-false-target associations. Hence, it is divided into three sub-matrices. The cells of the first $M_k \times N_{TGT}$ sub-matrix contain the scores for assigning measurements to existing tracks. The second $M_k \times M_k$ sub-matrix accounts for classifying measurements as new targets. The last $M_k \times M_k$ sub-matrix accounts for classifying measurements as false alarms. This matrix corresponds one-to-one to the set of all posterior hypotheses of Ω_i^k in the manner of Reid. A proof for that can be found in [DN06].

There are two remaining issues to be solved. Applying Murty's algorithm on L will not result in the n-best posterior hypothesis because:

1. the probabilities of the hypothesis rating equation are connected by multiplications, while assignment costs are the sums of matrix elements,
2. the underlying LAP solver minimizes the assignment costs which results in the n-worst assignments instead of the n-best.

It turns out that both problems can be eliminated by taking the negative logarithm of each matrix element. This expresses multiplications through additions (e. g. $\log(ab) = \log(a) + \log(b)$). Thus, it transforms the sum of matrix elements into multiplications. Furthermore, the negative logarithm maps probabilities to the inverse order of numbers which results in the n-best assignments: $-\log : (0,1] \mapsto [0,\infty)$. Consequently, the final logarithmic data association matrix $L^\star\left(\Omega_i^k\right)$ is defined as:

$$L^\star\left(\Omega_i^k\right) = \left(l^\star{}_{ij}\right) := \left(\begin{array}{ccc|ccc} -\log l_{11} & \cdots & -\log l_{1,N_{TGT}} & & & \\ \vdots & \ddots & \vdots & & & \\ -\log l_{M_k 1} & \cdots & -\log l_{M_k N_{TGT}} & & & \\ \hline & & & -\log \frac{\beta_{NT}(1-P_D)}{P_D} & \infty & \cdots \\ & & & \infty & \ddots & \infty \\ & & & \vdots & \infty & -\log \frac{\beta_{NT}(1-P_D)}{P_D} \\ \hline & & & -\log \frac{\beta_{FT}(1-P_D)}{P_D} & \infty & \cdots \\ & & & \infty & \ddots & \infty \\ & & & \vdots & \infty & -\log \frac{\beta_{FT}(1-P_D)}{P_D} \end{array}\right). \tag{4.21}$$

4.4.3.3 New Hypothesis Generation Algorithm

The last step to undertake in order to arrive at the new hypothesis generation algorithm is to combine the number of posterior hypotheses N_i^k, the logarithmic data association matrix $L^\star$ and Murty's algorithm.

To determine the set of the n-best new hypotheses Ω^{k+1}, given the set of all hypotheses Ω^k of the previous time-step, the cost matrix $L^\star$ for each hypothesis Ω_i^k has to be constructed first. Next, Murty's algorithm has to be applied on each cost matrix $L^\star(\Omega_i^k)$ to solve for the n-best posterior hypotheses ($\widehat{=}$ assignments). Finally, out of the new n^2 hypotheses the n hypotheses with highest probability have to be kept. The remaining $n^2 - n$ hypotheses have to be discarded. The pseudo-code for this procedure is given in algorithm 4.3.

Algorithm 4.3 The n-best MHT algorithm employing Murty's algorithm.

Input: Set of hypotheses Ω^k, set of measurements $Z(k+1)$, hypotheses count n
Output: Set of n-best posterior hypotheses Ω^{k+1}

1: **for** $\Omega_i^k \in \Omega^k$ **do**
2: Construct cost matrix $L^\star\left(\Omega_i^k\right)$
3: Determine set of n-best posterior hypotheses Ψ using Murty's algorithm
4: $\Omega^{k+1} = \Omega^{k+1} \cup \Psi$
5: **end for**
6: Sort Ω^{k+1} by hypothesis probability
7: $\Omega^{k+1} = \Omega^{k+1} \setminus \{\Omega_i^{k+1} : P_i^{k+1} > P_n^{k+1}\}$

It is important to note that in order to directly solve for the n-best hypotheses, still n^2 hypotheses have to be enumerated. The reason for this is that as no parent hypothesis probabilities are incorporated into the cost matrix, e. g. the 50$^{\text{th}}$ posterior hypothesis of Ω_1^k might have a smaller probability than the 1$^{\text{st}}$ posterior hypothesis of Ω_2^k. Therefore, all n-best posterior hypotheses of all prior hypotheses have to be enumerated in order to guarantee that the n-best are found.

4.4.4 Modified Gating

The gating technique, described in section 2.3, cannot be directly applied in n-best MHT. In n-best MHT all measurements have to be considered because a static cost matrix is used for data association, in which each cell has to be assigned to a meaningful value.

However, it turns out that it is possible to exclude non-gating measurements in a way that the computational workload is reduced considerably. The structure of the cost matrix in Murty's algorithm can be exploited to avoid the partitioning of certain nodes by assigning the respective matrix element of a measurement-to-track association to ∞, if the measurement does not fall into the gating region. This cell is henceforth *excluded* from the assignment.

4.5 Direct n-best Multiple Hypothesis Tracking

4.5.1 Introduction

In the n-best MHT approach, presented in section 4.4.3.3, most of the execution time is spent for solving LAPs while nodes are partitioned in Murty's algorithm. The number of LAPs to be solved is linear in the number of solutions to compute n [Mur68]. Hence, its complexity as a function of n is $\mathcal{O}(n)$. As for each of the n hypothesis n-best posterior hypotheses have to be determined, the resulting complexity of the n-best MHT approach as a function of n is $n\mathcal{O}(n) = \mathcal{O}(n^2)$.

It turns out that a considerable amount of computational costs can be avoided by directly initializing the node list U of Murty's algorithm with n nodes that contain modified cost matrices for all n hypotheses. The cost matrices are modified in such a way that the LAP assignment costs correspond one-to-one to Reid's hypothesis costs of equation (4.11). Hence, the parent hypothesis probabilities are incorporated into the assignment matrices, which makes a ranking based on the LAP assignment costs possible. This modification of the original n-best MHT approach is henceforth called the *direct-n-best MHT* approach.

In the next section the modified data association matrix is presented and it is proved that it corresponds one-to-one to the set of all feasible hypotheses in the manner of Reid. Finally, the new n-best MHT variant is given in pseudo-code.

4.5.2 Modified Data Association Matrix

The modified cost matrix $L'(\Omega_i^k) \in \mathbb{R}^{M_k \times (N_{TGT}+2M_k)}$ of a hypothesis Ω_i^k is is constructed in the following way. The elements of the cost matrix correspond directly to the three probabilities for track continuation, new target and false target associations of the original hypothesis probability equation (4.11) of Reid, derived in section 4.3.3. Given the probability for assigning a measurement $z_i(k+1)$, $1 \leq i \leq M_k$ to an existing track $1 \leq j \leq N_{TGT}$

$$l'_{ij} = \frac{P_D f_{\mathcal{N}\left(Hx_j^k, S_j^k\right)}(z_i(k+1))}{1-P_D}, \tag{4.22}$$

the modified data association matrix is constructed as:

$$L'(\Omega_i^k) = (l'_{ij}) = \sqrt[M_k]{P_{m(i)}^k (1-P_D)^{N_{TGT}}} \cdot \left(\begin{array}{ccc|ccc|ccc} l'_{11} & \cdots & l'_{1,N_{TGT}} & \beta_{NT} & 0 & \cdots & \beta_{FT} & 0 & \cdots \\ \vdots & \ddots & \vdots & 0 & \ddots & 0 & 0 & \ddots & 0 \\ l'_{M_k,1} & \cdots & l'_{M_k,N_{TGT}} & \vdots & 0 & \beta_{NT} & \vdots & 0 & \beta_{FT} \end{array}\right). \tag{4.23}$$

It is proved now that the modified data association matrix L' as well corresponds one-to-one to the set of all feasible data association hypotheses in the manner of Reid. In comparison to the previously presented data association matrix L the matrix L' encodes as well the parent hypothesis probability. The proof is analog to the one in the paper of Danchick and Newnam [DN06].

Theorem 2. *The product indexed over the assignment index pairs of the modified data association matrix L' corresponds one-to-one with the set of all hypotheses in the manner of Reid and equals the probability P_i^k of equation* (4.11)*.*

Proof. In each time-step k, N_{DT} of the M_k new measurements are assigned to existing tracks. These measurement-to-track assignments are indexed by the set of feasible assignment index pairs

$$\alpha = \{(m_1,n_1),\ldots,(m_{N_{DT}},n_{N_{DT}})\}.$$

Furthermore, N_{NT} of the M_k new measurements are classified as new targets and N_{FT} of the M_k new measurements are classified as false targets. Let the set of new and false target measurements be indexed by:

$$\beta = \{m_{N_{DT}+1}, \ldots, m_{N_{DT}+N_{NT}}\}, \text{ and}$$
$$\gamma = \{m_{N_{DT}+N_{NT}+1}, \ldots, m_{M_k}\}.$$

The allocation of the index sets is supposed to be compatible with the definition of feasible LAP assignments as defined in section 4.4.2.1.
Moreover, to fulfill the single source assumption, not more than one new measurement can be assigned to more than one target. Therefore, the set of index pairs

$$\delta = \alpha \cup \{(r, r + N_{TGT}) | r \in \beta\} \cup \{(s, s + N_{TGT} + M_k) | s \in \gamma\}$$

is a feasible assignment for L' which accounts for continued, new and false targets. It can be shown that the product over all matrix elements contained in the index pair set δ is equal to the hypothesis probability due to Reid.

$$\begin{aligned}
\prod_{(t,u)\in\delta} {l'}_{tu} &= \sqrt[M_k]{P^k_{m(i)}(1-P_D)^{N_{TGT}}}^{N_{DT}+N_{NT}+N_{FT}} \left(\prod_{(m,n)\in\alpha} {l'}_{mn} \right) {\beta_{NT}}^{N_{NT}} {\beta_{FT}}^{N_{FT}} \\
&= P^k_{m(i)}(1-P_D)^{N_{TGT}} \left(\prod_{i=1}^{N_{DT}} \frac{P_D f_{\mathcal{N}}(z_{m_i}(k+1))}{1-P_D} \right) {\beta_{NT}}^{N_{NT}} {\beta_{FT}}^{N_{FT}} \\
&= P^k_{m(i)} \frac{(1-P_D)^{N_{TGT}}}{(1-P_D)^{N_{DT}}} \left(\prod_{i=1}^{N_{DT}} P_D f_{\mathcal{N}}(z_{m_i}(k+1)) \right) {\beta_{NT}}^{N_{NT}} {\beta_{FT}}^{N_{FT}} \\
&= P^k_{m(i)} {P_D}^{N_{DT}} (1-P_D)^{N_{TGT}-N_{DT}} \left(\prod_{i=1}^{N_{DT}} f_{\mathcal{N}}(z_{m_i}(k+1)) \right) {\beta_{NT}}^{N_{NT}} {\beta_{FT}}^{N_{FT}} \\
&= P_i^{k+1} \text{ (from equation 4.11)}
\end{aligned}$$

□

The idea behind this construction is that any assignment returned by Murty's algorithm consists of M_k measurement-to-target assignments because each measurement has to be assigned to a target. Therefore, the total hypothesis probability in the manner of Reid can be obtained by multiplying each matrix element by $\sqrt[M_k]{P^k_{m(i)}(1-P_D)^{N_{TGT}}}$. The square root disappears because exactly M_k factors are multiplied and the remaining factor $P^k_{m(i)}(1-P_D)^{N_{TGT}}$ incorporates the probability of the parent hypothesis into the

assignment costs.

As the probabilities of the hypothesis rating equation are connected by multiplications, while assignment costs are the sums of matrix elements, the negative logarithm cost matrix $L'^{\star}$ has to be constructed analogously to $L^{\star}$ in the n-best MHT variant (see section 4.4.3.2).

Finally, the node list U of Murty's algorithm has to be initialized and Murty's algorithm can be applied to directly solve for the n-best hypotheses.

4.5.3 Modified Application of Murty's Algorithm

After the negative logarithm of each matrix element is computed, Murty's algorithm is applied to solve for the n-best assignments. However, in contrast to the application of Murty's algorithm in the n-best MHT approach the list of nodes U is initialized differently. In the beginning it is filled with the nodes of all modified data association matrices $L'^{\star}\left(\Omega_i^k\right)$ for each hypothesis Ω_i^k. The rest of the algorithm works as before. Thanks to the modification of the data association matrices each assignment cost corresponds exactly to the hypothesis probability the assignment encodes. Hence, the n-best posterior hypotheses can be determined by just partitioning n nodes instead of n^2. Though, n LAPs have to be solved in the initialization step. The pseudo-code for the direct n-best MHT variant is given in algorithm 4.4 below.

Algorithm 4.4 The direct n-best MHT algorithm.

Input: Set of hypotheses Ω^k, set of measurements $Z(k+1)$, hypotheses count n
Output: Set of n-best posterior hypotheses Ω^{k+1}

1: **for** $\Omega_i^k \in \Omega^k$ **do**
2: Construct cost matrix $L'^{\star}\left(\Omega_i^k\right)$
3: Determine best posterior hypothesis of $L'^{\star}\left(\Omega_i^k\right)$ and construct node N_i
4: $U \leftarrow U \cup \{N_i\}$
5: **end for**
6: Determine set of n-best hypotheses Ω^{k+1} by applying Murty's algorithm on U

It should be mentioned explicitly that the **while**-loop of Murty's algorithm is dragged out

of the **for**-loop running over the set of hypotheses. This reduces the overall algorithmic complexity as a function of the number of maintained hypotheses from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

4.6 Summary

In the previous sections the theory required to understand a state-of-the-art multiple hypothesis tracker was presented. The chapter covered Reid's hypothesis-oriented, original formulation of the MHT algorithm which consists of an iterative tree-based hypotheses generation method and an equation to rate hypotheses. Moreover, certain practical issues of the conceptional MHT could be solved with the help of different pruning strategies. However, the inherent exponential algorithmic complexity could only be solved by reformulating the whole method as a LAP and exploiting Murty's algorithm to solve directly for the n-best posterior hypotheses. The original n-best MHT variant of Danchick and Newnam was improved further. Its complexity could be reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. The n-best MHT variants avoid generating, sorting and pruning all possible hypotheses from scan to scan. Thus, they take off the exponential character from Reid's original MHT approach. To be more precise: previously intractable MTT problems can now be solved in real-time. A detailed evaluation of the different MHT formulations and optimizations is given in the evaluation chapter. An actual prototype of a full-featured multiple hypothesis tracker is presented first in the following chapter.

5 Implementation

In this chapter the MHT implementation which was developed for the purpose of this master thesis is presented. First, some practical issues that arose while working on the implementation are discussed. They are independent of any theoretical consideration but crucial for the implementation. Second, it is outlined how the n-best MHT hypothesis generation algorithm was optimized further for speed. Lastly, the tracking software, the used programming languages and libraries as well as the source code are presented.

5.1 Practical Issues

While implementing the MHT algorithm a number of practical issues arose. In this section the most striking ones are discussed.

5.1.1 Track Management

In comparison to the standard implementations of JPDAF or GNNT which can handle only a fixed and previously known number of targets, MHT naturally supports track initiation and track termination.
As every single new measurement is interpreted as a new target within the hypothesis generation process of Reid, it is important to classify tracks as *tentative* or *confirmed*. Otherwise, the tracker output would be flooded with numerous tracks that exist only for one time-step.

The classification can be performed on the basis of the number of consecutive track updates. If a certain track was updated $n \in \mathbb{N}$ times in a row, it is regarded as existing and its life stage is switched from tentative to confirmed.

Old life stage	New life stage	Rules
-	Tentative	Every newly created track.
Tentative	Confirmed	After n successive track updates.
Confirmed	Deleted	As soon as the biggest eigen-values of the state covariance matrix of a track exceeds a threshold, or the track was not updated for m or more time-steps.
Tentative	Deleted	Track was not updated for m or more time-steps.

Table 5.1: Transition rules for all track life stages.

Tracks that disappeared need to be deleted. Detecting whether a certain track disappeared can be performed on the basis of its state covariance matrix. If the biggest eigen-value λ_{max} (describing the track uncertainty) of it exceeds a certain threshold B, the life stage of the track is switched from either tentative or confirmed to *deleted*.
Another possibility is to count for how many successive iterations the track was not updated. When the number of iterations in which the track was not updated exceeds a certain threshold, the life stage of the track is switched to deleted. In table 5.1 the transition rules for the different track life stages are summarized.

5.1.2 Track Trees

For a solid implementation it is crucial to exploit the available memory and computing power as well as possible. Efficient data structures for the hypothesis management have to be used because potentially a large number of intermediate hypotheses has to be generated to find the n-best ones. Keeping duplicates of many tracks within different hypotheses significantly increases the memory footprint as well as the execution time. Therefore, an often applied implementation technique among the MHT community is *track trees*, also known as *family structure* [Kur90, Bla04]. This MHT implementation approach is known as *track-oriented* MHT.
In track-oriented MHT a track tree is created for each postulated target. The root of each track tree represents its birth and its branches represent continuations of the track by different measurements. A trace of successive branches from the root to a leaf represents one particular track. Then, every hypothesis only has to store pointers to the individual

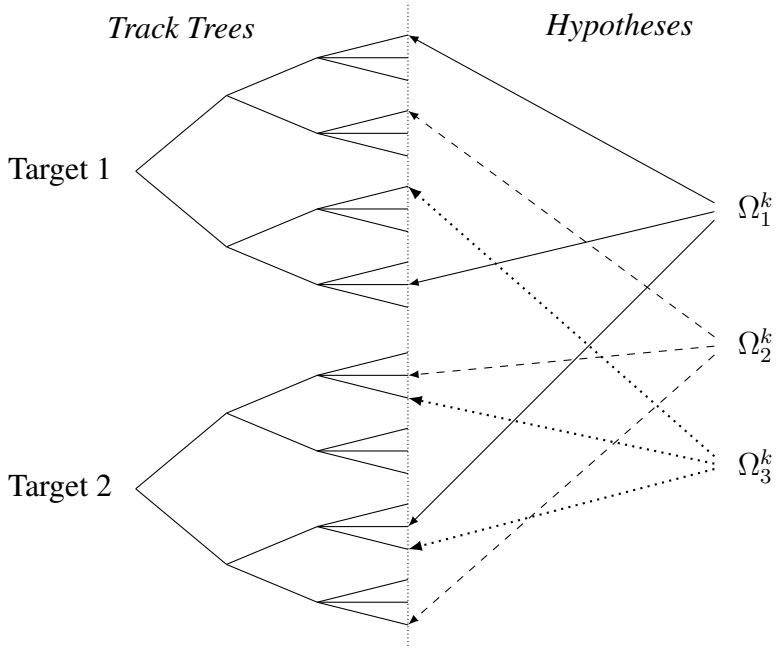


Figure 5.1: Relation between track trees (*track hypotheses*) and global hypotheses

track tree nodes and not entire, potentially duplicated, tracks. Hence, track tree nodes are also called *track hypotheses*. In figure 5.1 track trees and their relation to the *global hypotheses* is illustrated.

The main difficulty in the implementation of track trees is the deletion of a track tree node when there is no global hypotheses *referencing* that node anymore. Therefore, it is important to keep track of the global hypotheses and the track tree nodes that reference other track tree nodes. As soon as the reference count of a node reaches zero that node can be deleted safely. The preferred data structure for implementing such a behavior is a reference counting system [Sch04].

5.1.3 Log-Likelihoods

The MHT algorithm performs complex probability calculations on ever shrinking hypothesis likelihoods. Due to the limited numerical precision of the floating-point arithmetic of today's computers it is unfeasible to work directly with probabilities of the interval $[0,1]$. Taking the logarithm of a probability $p \in (0,1]$ maps the small range $(0,1]$ to $(-\infty,0]$. This reduces significantly numerical precision problems. Such likelihoods are

naturally called *log-likelihoods*.
An additional advantage is that complex arithmetic operations become faster and numerically more stable. For example, multiplications turn into additions $\log(ab) = \log(a) + \log(b)$ and exponentiations turn into multiplications $\log(a^b) = b\log(a)$. Especially, the latter property only makes the calculation of hypothesis probabilities feasible because the exponentiation can be dragged out of the logarithm. This is particularly important for very big and very small b.

5.1.4 A Solver for the Linear Assignment Problem

The JVC algorithm for the rectangular case was chosen because it is generally accepted to be the fastest method to solve dense RLAPs [PPBS99]. An open-source implementation was used because the focus of this thesis lies on the implementation of the MHT algorithm and not on the RLAP solver. The implementation supplied with the book "Assignment Problems" by Burkard, Dell'Amico and Martello [BDM09] was used. The source code can be found on the accompanying web-site[1]. It had to be ported to C++ as the original version was written in Pascal.

5.1.5 Painting Gating Ellipses

Obtaining a visual feedback of the tracking process is beneficial. It facilitates understanding what exactly happens in certain tracking situations and thus, it helps to find potential bugs. While it is easy to visualize tracks, visualizing gating regions is not that straightforward. The gating region of a track is defined by a covariance matrix describing uncertainty. The gate has the shape of a p-dimensional *hyper ellipsoid* where p is the dimension of the underlying Gaussian distribution. In this section it will be derived how to draw 2-dimensional gating ellipses on the basis of a covariance matrix $A \in \mathbb{R}^{2\times 2}$. The resulting size and orientation describe the *level set* of the Mahalanobis distance $d_M(x,c,A)$. Hence, the fundamental problem is to extract the width, height and angle of such an *error ellipse* given a covariance matrix A.

A 2-dimensional circle with radius $r \in \mathbb{R}_+$ centered at the origin can be described as

[1] `http://www.assignmentproblems.com/LAPJV.htm`.

$|x| = r \Leftrightarrow xx^T = r^2$. Applying a scale on both *half-axes* equals the multiplication of x by a scaling matrix $S = \begin{pmatrix} \frac{1}{s_x} & 0 \\ 0 & \frac{1}{s_y} \end{pmatrix} \in \mathbb{R}^{2\times 2}$:

$$(Sx)^T(Sx) = x^T(S^TS)x = x^TS^2x = r^2 \;\; (S \text{ diagonal} \Rightarrow S^T = S)$$

Plugging in $x = (a\ b)^T$ expands this equation into $\frac{a^2}{{s_x}^2} + \frac{b^2}{{s_y}^2} = r^2$. Then the horizontal and vertical radii r_x and r_y can be determined. By setting $x = (r_x\ 0)^T$ it follows that $\frac{{r_x}^2}{{s_x}^2} + \frac{0^2}{{s_y}^2} = r^2 \Leftrightarrow r_x = s_xr$, respectively $r_y = s_yr$.
To allow arbitrary orientations x has to be additionally multiplied by a rotation matrix $R = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \in \mathbb{R}^{2\times 2}$. This yields the expression

$$(R^{-1}x)^TS^2(R^{-1}x) = x^TRS^2R^Tx = r^2 \;\; (R \text{ orthogonal} \Rightarrow R^{-1} = R^T).$$

To move the midpoint of the ellipse to an arbitrary position its center $c \in \mathbb{R}$ is subtracted from x.

$$(x-c)^T \underbrace{RS^2R^T}_{=:A\in\mathbb{R}^{2\times 2}}(x-c) = r^2 \tag{5.1}$$

The matrix A is *symmetric* and *positive semi-definite*. It describes a scaled, rotated and translated ellipse. Given such a matrix A, the aim is to decompose it into R and S in order to solve for the horizontal radius r_x, the vertical radius r_y and the angle ϕ. This can be achieved by performing an *eigen decomposition*. The eigen decomposition factors a positive semi-definite matrix A into $A = Q\Lambda Q^{-1}$. In that expression Q corresponds to R and Λ to S^2. The matrix Λ contains the eigen values $\lambda_{x,y}$ of A on its diagonal.

With $S^2 = \begin{pmatrix} \frac{1}{{s_x}^2} & 0 \\ 0 & \frac{1}{{s_y}^2} \end{pmatrix} = \begin{pmatrix} \lambda_x & 0 \\ 0 & \lambda_y \end{pmatrix}$ it follows that $s_x = \frac{1}{\sqrt{\lambda_x}}$ and $s_y = \frac{1}{\sqrt{\lambda_y}}$ and therefore the radii can be calculated as:

$$r_x = \frac{r}{\sqrt{\lambda_x}} \text{ and } r_y = \frac{r}{\sqrt{\lambda_y}}. \tag{5.2}$$

With $R = Q$ it follows that the angle ϕ can be calculated as:

$$\phi = \cos^{-1}(q_{11}). \tag{5.3}$$

This result can now be applied on the Mahalanobis distance $d_M(x,\mu,\Sigma)$ which represents the level set of a Gaussian distributions $\mathcal{N}_2(\mu,\sigma)$. The mean-value μ is used as the ellipse center c and its positive semi-definite covariance matrix Σ as the ellipse matrix A.

However, it should be mentioned that the covariance matrix Σ has to be inverted before it can be plugged into the above equations because the Mahalanobis distance is defined like this.

5.2 Software Optimization

The aim of this thesis is to create a *real-time capable* MHT implementation. Hence, the speed of the final implementation plays a major role for its overall quality. To speedup a software different steps are taken in a specific order to arrive at a fast implementation.

1. A simple but working version should be always implemented first because in the majority of cases most of the time is spent in parts of the code which the programmer would not expect[2].

2. After that, the software can be optimized. A *profiler* (like Intel VTune[3] or the minimalistic but very effective profiler Very Sleepy[4]) is used to find out in which parts of the code most of the execution time is spent[5].

3. After the execution *hotspots* are identified, they are optimized. First, high-level optimizations as replacing data structures are applied. Only then, low-level optimizations as linearizing memory accesses are applied.

In the MHT implementation the Munkres LAP solver was first replaced by the faster JVC algorithm. The second optimization step was to verify if the data structures were used appropriately. The used C++ Standard Library containers were adapted thoroughly so that they are ideally suited for the given problems. Next, memory usage and memory access patterns were optimized. Today, in most of the software memory access constitutes the major performance bottleneck. Difficulties in cache usage, arising from non-sequential memory access patterns, forces the CPU to access the main memory frequently. Therefore, the number of reallocations per scan and the data structure sizes were minimized.

[2] "We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil." (Donald Knuth in [Knu74])

[3] `http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe`

[4] `http://www.codersnotes.com/sleepy`

[5] In software engineering it is said that in most cases 90% of the time is spent in 10% of the code (known in that context as the *90/10 law*).

The remaining major bottleneck was the usage of Murty's algorithm in the n-best MHT variant. It turned out that by slightly modifying the data association matrix the speed could be increased dramatically (see section 4.5).

5.3 Tracking Software

In this section the tracking software and its source code are presented. The software consists of two independent modules: the tracker and the simulation creator as first presented. It is followed by a short overview of the source code and the used libraries.

5.3.1 Simulation Creator

The tracking software consists of two tabs. On the first tab the tracker is located. On the second tab the simulation creator is situated. The simulation creator can be used to quickly create synthetic tracking scenarios. An arbitrary number of tracks with position, start time and duration can be easily created with a few mouse clicks. Additional parameters regarding the tracking environment and the sensor can be specified as well. The target density parameters β_{NT} and β_{FT} are calculated automatically. In figure 5.2 the simulation creator is depicted. The parameters required for creating a simulation are listed in the following table.

Scan volume of sensor	V
Measuring interval of sensor	J
Detection probability of sensor	P_D
Measurement noise	R
Average clutter count per scan	λ

Table 5.2: Parameters required for creating a simulation.

The measurement noise is determined using two independent normal distributions for the x- and y-coordinates. The number of cluttered measurements follows a Poisson distribution and their locations are uniformly distributed over the scan volume. The probability that a measurement is detected is uniformly distributed is as well.

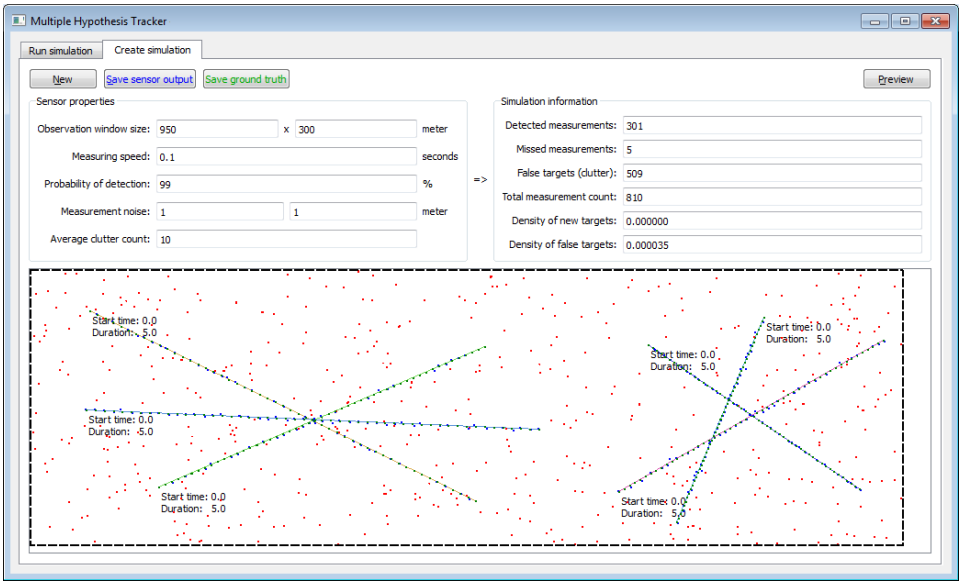


Figure 5.2: The user-interface of the simulation creator. A few crossing targets and the corresponding sensor output (disturbed target measurements and clutter) are depicted.

By clicking the "'Preview"' button on the upper right the targets are sampled, clutter is generated for each scan and the resulting dataset is painted. Clutter is painted in red, disturbed measurements originating from a target output by the sensor in blue and ground truth measurements in green.

The sensor output and the ground truth can be saved as CSV[6] files by clicking on the two buttons in the upper left. A sensor output CSV file contains one line per measurement and each line consists of `(scan number, measurement position x, y)`. A ground truth CSV file contains one line per measurement as well, but each line additionally contains the label of the target to which the measurement belongs: `(scan number, target label, measurement position x, y)`.

5.3.2 Tracker

In the first tab of the software the tracker is located. With the buttons in the upper right a CSV file containing the sensor output can be loaded. This data is fed into the MHT

[6]The *Comma Separated Values* (CSV) file format stores tabular data as plain-text files. Each line in a CSV file is a record and each record is divided into multiple fields, separated by a special character. Most commonly this is a tab or a comma.

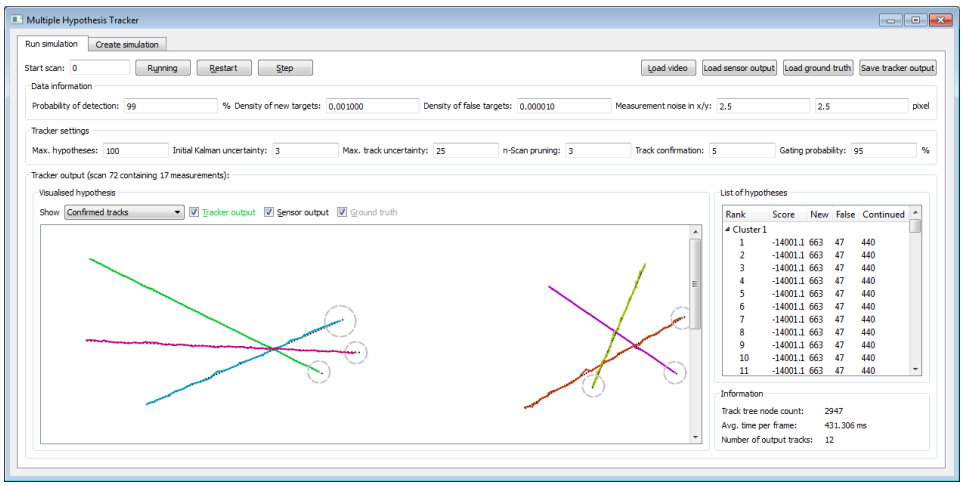


Figure 5.3: The tracks estimated when tracking synthetic data.

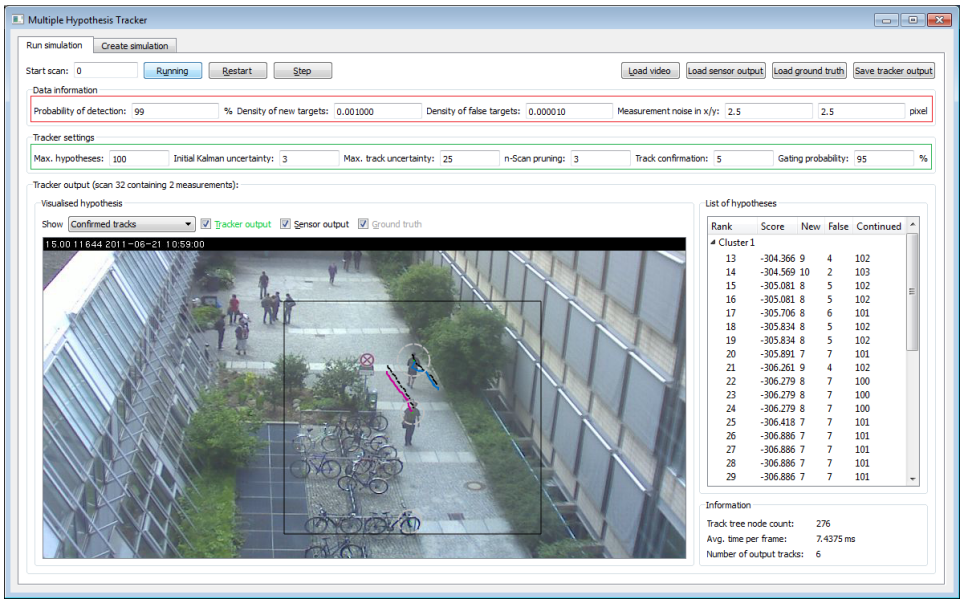


Figure 5.4: The tracks estimated when tracking real-world data.

algorithm and the resulting estimated tracks are painted into the bottom view. Additionally, a CSV file containing the ground truth and a source video can be loaded. If provided, they are painted into the bottom view as well. For synthetic data no video is available. The tracking process can be started, paused and continued step-by-step by using the buttons in the upper left. The tracker applied on synthetic data and real-world data is depicted in figure 5.3 and 5.4.

Several parameters have to be provided in order to properly track targets in the sensor

output. Some of these parameters depend on the tracking scenario. They are entered in the first row of parameters (marked in red in figure 5.4). These parameters are the detection probability P_D, the new target density β_{NT}, the false target density β_{FT} and the measurement noise R. The second set of parameters configures certain tracker settings. They are entered in the second row of parameters (marked in green in figure 5.4). These parameters are the maximum number of hypotheses, the initial Kalman filter uncertainty, the maximum track uncertainty, the threshold for n-scan pruning, the track confirmation count and the gating probability P_G. In the bottom group box the tracking output is shown. The current scan number, as well as how many measurements it contains is displayed. Moreover, it can be chosen which tracks in which life stage are painted: tentative and confirmed, only confirmed, only tentative or deleted tracks. Additionally, it can be specified by using the three check boxes if the tracking output, the sensor output and the ground truth are painted or not.

5.3.3 Source Code

The MHT algorithm, the simulation creator and the tracker were entirely implemented in C++. This choice was made because even nowadays C++ is still the most widely used programming language for object-oriented high performance software development.
Additionally, heavy usage of the *C++ Standard Library* was made and the *Eigen*[7] library was used for matrix and vector algebra. The user-interface was created with the aid of the *Qt* framework[8].

The MHT algorithm is implemented in a customizable manner by employing templates in order to use vectors and matrices of arbitrary size that do not require any dynamic heap allocations. The dimensions of the measurement vectors D_M and of the state vectors D_S have to be specified when a variable of the main class of the tracker implementation is declared: `MHT::Tracker<DS,DM>`. Hence, all state and measurement vectors and all covariance and noise matrices have a static size which is known at compile time. Still, the data association matrices in Murty's algorithm are of unpredictable dimensions. Here, the Eigen library comes in handy. It provides matrices of dynamic size which do not need any heap allocations but expect a maximum size argument (e. g. `Eigen::Matrix<float,`

[7] `http://eigen.tuxfamily.org/index.php?title=Main_Page`
[8] `http://qt.nokia.com/`

`Eigen::Dynamic, Eigen::Dynamic, Eigen::ColMajor, MAX_ROWS, MAX_COLS>`).

The source code of the implementation is distributed over a number of header files directly implementing terms from theory, such as a *track* or a *hypothesis*. As templates are used, the implementation has to be put into the header files. The reason for that is that the implementation of a class has to be accessible for the compiler when it instantiates the class during compile time.
In figure 5.5 the structure of the whole source code is illustrated in the form of an include dependency graph. All MHT algorithm classes are encapsulated in a namespace called `MHT`. The base class providing the user with all functionality is called `Tracker<DM,DS>`. All the necessary steps to undertake are to pass the set of measurements in each new scan by using the `Tracker::update()` function and to retrieve the most probable hypothesis for the current scan by using the `Tracker<DM,DS>::mostProbableHypothesis()` function. The `Tracker <DM,DS>` class maintains a set of clusters encapsulated in the class `Cluster<DM,DS>`. Each cluster holds a set of hypotheses. A single hypothesis is implemented in the `Hypothesis <DM,DS>` class. A hypothesis is a collection of tracks encapsulated in the `Track<DM,DS>` class. A track holds a pointer to a track tree node encapsulated in the `TrackTreeNode<DM, DS>` class. Track tree nodes contain all information that is needed to manage tracks; e. g. the Kalman filter, which is implemented in the `KalmanFilter<DM, DS>` class. Additional utility classes and functions can be found in the `utils.hpp` header file. The user-interface is implemented in the `MainWnd` class. The view to paint targets of the simulation creator is implemented in the class `PaintView`. The whole user-interface design was created using the QtDesigner. It can be found in the file `mainwnd.ui`.

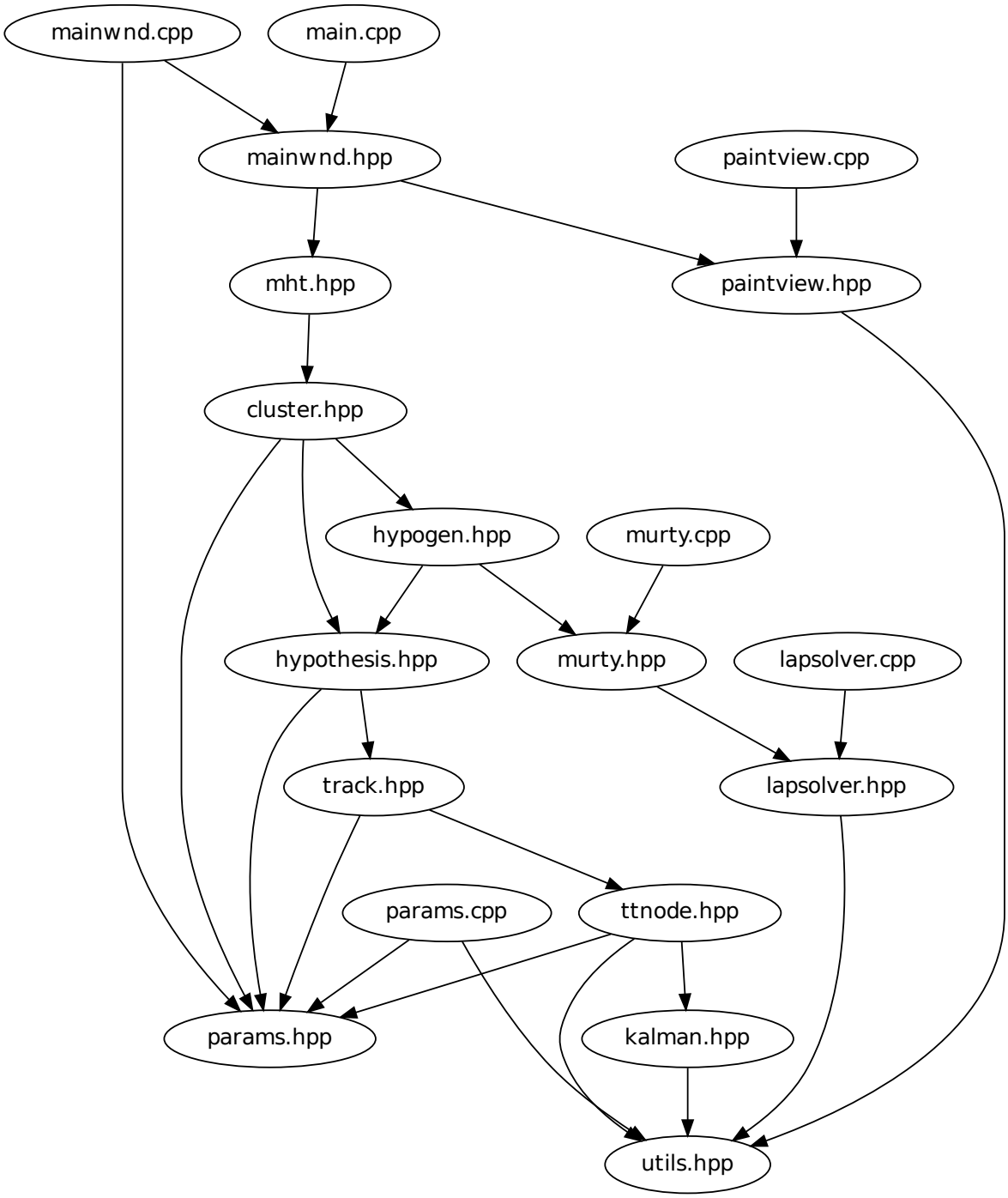


Figure 5.5: The source code structure in the form of an include dependency graph.

6 Evaluation

The implementations of the previously presented MHT variants are evaluated in this chapter. In the first part, fundamental principles behind the performance measurement of MTT systems are discussed and two metrics for measuring tracking performance are presented. In the second part, a practical evaluation using these metrics is performed. It is based on synthetic data on one hand and detections provided by a person detector applied to a real-world surveillance video on the other hand.

6.1 Performance Measurement of MTT Systems

While performance measurement of STT systems is well understood, measuring performance of MTT systems is a challenging task. In the following sections, the fundamentals of MTT performance measurement are outlined and the *CLEAR MOT* metrics for evaluating the tracking quality of MTT algorithms are presented.

6.1.1 Introduction

Nowadays, a systematic performance analysis is a crucial part in the development of any tracking system. It is needed, for instance, to measure the effects of optimizations made on a tracking algorithm or to compare the performance of different tracking approaches.

Performance measurement of STT systems is straightforward. There exist various tools like Euclidean distance, least-squares minimization or root-mean-square deviation that are used to measure tracking performance of STT algorithms. It turns out that it is difficult to come up with similar metrics for MTT. Even though, a wide range of MTT performance metrics exist today (see e. g. [GTK11, PHS+06, RVCV11]), there is still no

consensus on a general procedure of how to measure the performance of MTT systems [HM02]. The difficulty is that the number of targets in the ground truth G very likely differs from the number of tracks in the tracker output X. Finding a *correspondence* (assignment) between the two sets, even in cases where their cardinality is the same, is the first problem. An established correspondence already enables to calculate a *localization error*. Additionally, *configuration errors* such as *false positives*, *misses* (in cases of unequal cardinality) and *mismatches* (when the labels of tracks switch between successive frames) have to be considered. Coming up with just a few intuitive, expressive metrics that capture the aforementioned properties well is the major problem. At the same time, the number of used metrics should be as little as possible. Otherwise, making concrete comparisons of metric results is unfeasible because too many metrics have to be considered.

Furthermore, *frame-based* and *track-based* MTT performance metrics are distinguished [PHS+06]. The former one considers the tracking results of each frame independently by inspecting the track states and the track labels (e. g. see [PHS+06, GTK11]). In the latter type of metrics entire tracks are inspected at the same time which facilitates capturing temporal aspects (e. g. see [RVCV11]).

6.1.2 The CLEAR MOT Metrics

The *CLEAR MOT metrics* (2008) are two frame-based metrics that are used to evaluate MTT algorithms [BS08]. One metric, called *MOTP*, measures the *localization error* between the tracker output and the ground truth. The other metric, called *MOTA*, accounts for *configuration errors* (mismatches, misses and false positives).

6.1.2.1 Overview

First of all, the correspondence between the ground truth and the tracker output has to be established. In each time-step k a MTT system outputs a set of track states

$$X^k = \{x_1^k, x_2^k, \ldots\} \subset \mathbb{R}^{D_S}. \tag{6.1}$$

The corresponding ground truth contains the target states

$$G^k = \{g_1^k, g_2^k, \ldots\} \subset \mathbb{R}^{D_S}. \tag{6.2}$$

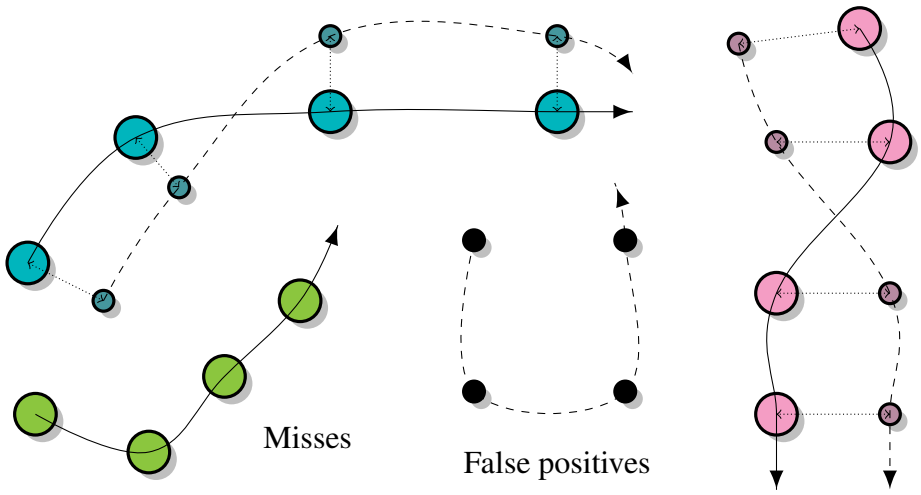


Figure 6.1: An illustration of the issues that can arise when associating the tracker output to the ground truth. Big circles are states of the ground truth. Small circles are states of the tracker output. The localization error between any two states is their Euclidean distance [BS08].

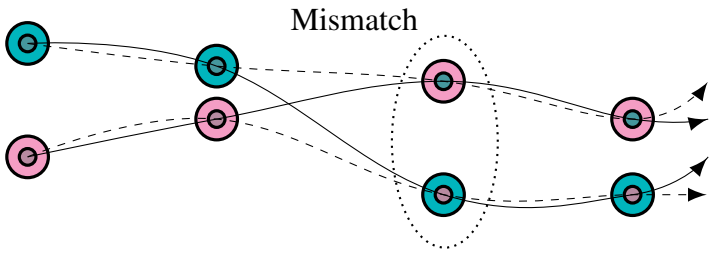


Figure 6.2: An illustration of a mismatch that can occur when tracking dense targets [BS08].

The elements of both sets consist of a track label and a state vector. It should be mentioned that the number of estimated states $|X^k|$ and the number of ground truth states $|G^k|$ do not have to be equal. In figure 6.1 the correspondence establishment and the terms localization error, miss and false positive are illustrated. In figure 6.2 a mismatch is depicted.

The general procedure of calculating the CLEAR MOT metrics MOTP and MOTA consists of the steps summarized below [BS08].

1. Establish the best correspondence between the states in the tracker output X^k and the states in the ground truth G^k for every time-step k.

2. For each assignment (i,j) between a tracker output state x_i^k and a ground truth state g_j^k, calculate the localization error d_i^k using e. g. the Euclidean distance.

3. Accumulate all configuration errors:

   a) Count all targets in the ground truth for which no track state was output by the tracker as *misses* $m_k \in \mathbb{N}_0$.

   b) Count all states in the tracker output for which no target in the ground truth exists as false positives $fp_k \in \mathbb{N}_0$.

   c) Count all occurrences where the track label of a state in the tracking output changes between two successive time-steps as mismatches $mme_k \in \mathbb{N}_0$.

6.1.2.2 Correspondence Establishment

The establishment of the correspondence between the tracker output and the ground truth is the central aspect of the CLEAR MOT metrics. To establish the correspondence for a time-step k a LAP (see section 4.4.2) is solved which assigns the ground truth to the tracker output states by minimizing the overall distance. This can be done e. g. with the use of the Hungarian method (see section 4.4.2.2) [FD91, DF92].
Naively, assigning each ground truth state to a tracker output state and counting all remaining ground truth states as misses and all remaining tracker output states as false positives sounds reasonable. Though, in practice, a few pitfalls have to considered.

First, a threshold $T_{MOT} \in \mathbb{R}_+$ is defined which is the maximum distance between the ground truth and a the tracker output states to be considered as potential matches. This is of great importance because big distances in most of the cases signify that the tracker missed the target and it tracks something else than a reasonable assignment. Hence, a correspondence between $x_i^k \in X^k$ and $g_j^k \in G^k$ is valid only if

$$\text{dist}\left(x_i^k, g_j^k\right) \leq T_{MOT}. \tag{6.3}$$

The value of T_{MOT} depends on the tracking problem and the used distance metric. Hence, it cannot be specified for the general case.
Second, there has to be a way to detect if a track label between two successive time-steps changed in order to account for configuration errors. Mismatches can occur when a track

is lost and when after a few scans it is reinitialized with a different label, or when two tracks get very close and the tracker in consequence mistakenly swaps their labels. To detect when mismatch errors occur, a list of mappings between ground truth target labels and tracker output track labels is maintained. Whenever a new association (i,j) is made which contradicts a previously made one (i,k) a mismatch is counted and (i,k) is replaced by (i,j). Otherwise, (i,k) is directly added to the list of mappings. Mismatch errors are counted only once in the time-step they occur. Especially in scenarios where many targets are subject to tracking and mismatches are frequent, this gives a more intuitive measure.

After the correspondence between the tracker output and the ground truth is established the MOTA and MOTP metrics can be calculated.

6.1.2.3 Performance Metrics

1. The *Multiple Object Tracking Precision* (MOTP) metric is the total localization error for all matched pairs of states from the tracker output and the ground truth averaged over all frames by the total number of made matches. $c_k \in \mathbb{N}_0$ denotes the number of correspondence matches in time-step k.

$$\text{MOTP}(G,X) := \sum_k \sum_i \frac{d_i^k}{c_k} \tag{6.4}$$

   The MOTP metric indicates how well a MTT system estimates the target positions. The metric does not account for any configuration errors (misses, false positives and mismatches). They are captured by a second metric described in the next paragraph.

2. The *Multiple Object Tracking Accuracy* (MOTA) metric captures the configuration errors that a MTT system makes averaged over all frames. It reflects how well a MTT system detects the targets and how well it follows their trajectories over time, independently of the quality of their position estimations.

$$\text{MOTA}(G,X) := 1 - \frac{\sum_k (m_k + fp_k + mme_k)}{\sum_k g_k} = 1 - \underbrace{(\overline{m} + \overline{fp} + \overline{mme})}_{=:E_{\text{tot}}} \tag{6.5}$$

The MOTA metric is the result of combining the following three ratios (each of them captures a different configuration error): the ratio of missed measurements

$$\overline{m} := \frac{\sum_k m_k}{\sum_k g_k}, \tag{6.6}$$

the ratio of false positives

$$\overline{fp} := \frac{\sum_k fp_k}{\sum_t g_k} \tag{6.7}$$

and the ratio of mismatches computed over the total number of measurements present within all frames

$$\overline{mme} := \frac{\sum_k mme_k}{\sum_k g_k}. \tag{6.8}$$

$g_k \in \mathbb{N}_0$ denotes the number of targets present in the ground-truth in time-step k.

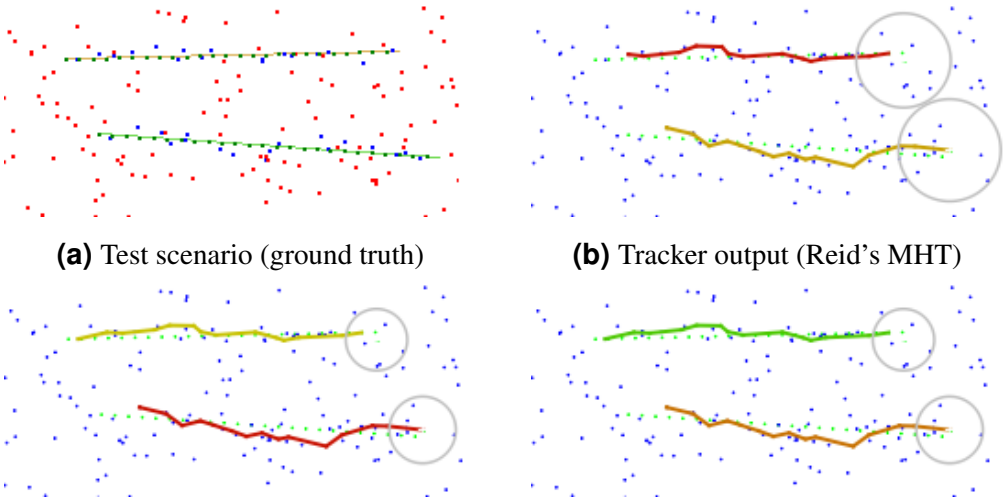
6.2 Practical Evaluation

In the following sections the practical evaluation of the MHT implementation on the basis of *synthetic* and *real-world* test scenarios is presented.
Synthetic test scenarios are suitable for evaluating algorithmic aspects because the tracking scenario, the sensor settings (e. g. clutter rate λ or detection probability P_D) and the tracking parameters (e. g. measurement noise covariance R) can be changed easily by using the simulation creator. However, it is of great importance to evaluate the MHT implementation as well in comparison to a real-world test scenario. Only that can show the practical side of the tracker in the presence of real-world effects such as missing measurements (e. g. caused by occlusion) or maneuvering targets.

The evaluation of synthetic tracking scenarios focus on the analysis of the speed of the algorithm, as well as on the tracking quality for pathological tracking scenarios. The real-world scenario is evaluated in order to see how well the MHT algorithm performs on a real dataset. As no results of other tracking algorithms exist for comparison, a simple GNNT (see section 3.3.1) was implemented additionally.

For the entire evaluation a computer with an AMD Phenom II X4 965[1] (3.4 GHz) CPU and 4 GB of main memory was used.

[1]The AMD Phenom II X4 965 CPU has 6 MB of L3 cache, 4 MB of L2 cache and 4 cores.



(a) Test scenario (ground truth)

(b) Tracker output (Reid's MHT)

(c) Tracker output (n-best MHT)

(d) Tracker output (direct n-best MHT)

Figure 6.3: The tracking scenario which is used to test the three MHT variants for equality and the tracker output of the three MHT variants.

6.2.1 Synthetic Scenarios

6.2.1.1 Equality of MHT Variants

In the previous chapters it was proven that Reid's MHT and the two n-best MHT variants compute the identical sets of hypotheses. However, in the beginning of the evaluation the equality of the output of the three approaches is verified in order to be sure of the correctness of the implementations. For this purpose a simple tracking scenario was created and the three algorithms were applied on it concurrently. The intermediate, pruned sets of hypotheses, the visual tracking output and the CLEAR MOT metrics were compared for equality.

It turned out that the implementation of the three algorithms deliver identical tracking outputs for the scenario depicted in figure 6.3. The track colors are calculated based on the track labels. The track colors are different because the three MHT variants generate the n-best hypotheses in different ways. Thus, the number of intermediate tracks is different and so are the track labels.
It turned out that it can happen that the two n-best approaches generate slightly different sets of hypotheses. In rare cases hypotheses occur that have the same probability as in Reid's approach but different configurations. The reason for this is that the order in which

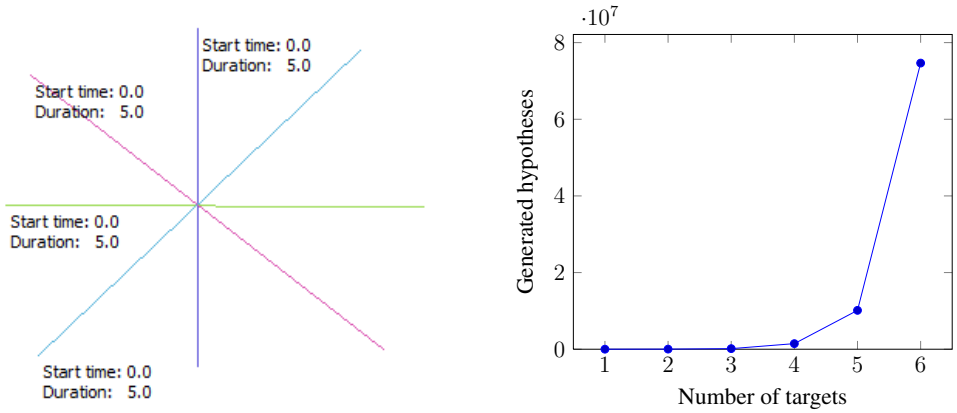
hypotheses are enumerated in the three approaches is completely different. Therefore, especially in the direct n-best approach the last few generated hypotheses might have the same probability as the ones of Reid's approach but different configurations. This is very likely to happen if $\beta_{NT} = \beta_{FT}$ because classifying a measurement as a new target has the same probability as classifying it as a false target. Hence, many hypotheses with equal probabilities are generated.

6.2.1.2 Tracking Speed

Implementing a real-time capable MHT system is the main task of this thesis. Therefore, improving the performance of Reid's original MHT algorithm is of major importance for it. Different optimizations were presented and implemented. In the following sections their speed and memory requirements as a function of different parameters are examined.

6.2.1.2.1 Conceptional MHT When comparing the different MHT optimizations and variants described in the previous chapters, it turns out that the original MHT version of Reid has to be considered independently. Its inherent exponential nature makes it inapplicable to any reasonable tracking problem. The problem is that the computation of the conceptional MHT algorithm quickly exceeds the available computing power and, even faster, the available amount of main memory of the used computer, causing an *out-of-memory* exception.
In this paragraph the exponential growth in the number of hypotheses going hand in hand with the growth of the amount of required memory and computing power is demonstrated. The tracking scenario for four targets is depicted in figure 6.4a. More and more targets that are equally distributed on the edge of a circle and that are moving towards its center are tracked. In figure 6.4b the number of generated hypotheses as a function of the number of tracked targets is shown. Even when clustering is enabled the given tracking scenario stays unfeasible because all tracks are crossing at one point and then they are combined into one super cluster. Updating this super cluster will still require the same maximum number of hypotheses, resulting in exactly the same graph as shown in figure 6.4b.

The original MHT variant of Reid is mainly of academic use because it is not applicable to most tracking scenarios. As soon as the number of targets exceeds 6, the max-



(a) Four circular arranged targets that move to the center of the circle.

(b) Number of hypotheses as a function of the number of targets.

Figure 6.4: The tracking scenario used to evaluate the growth in the number of maximally generated hypotheses as a function of the number of targets.

imum number of generated hypotheses exceeds the available memory of 4 GB. As a consequence also no analysis of the conceptional MHT variant in combination with clustering is performed.

6.2.1.2.2 n-best MHT Variants

To evaluate the n-best and direct n-best MHT variants the synthetic tracking scenario, depicted in figure 6.5, is used. This scenario contains 8 targets and no clutter. Hence, the number of measurements in each scan is 8 because for each of the 8 targets the sensor delivers one measurement per scan. To account for the clustering optimization, the tracking scenario consist of two independent groups of 4 targets. Such a simple test scenario was chosen because the required execution time for a more complex one is already too long for some MHT variants.

In the figures 6.6a, 6.6b and 6.7 the average number of generated hypotheses, the average update time and average number of solved LAPs are plotted as a function of the number of maintained hypotheses for different MHT variants. *Semi-logarithmic*[2] plots are used to make the large differences of the plotted values visible.
It can be clearly seen that the number of generated hypotheses in the n-best MHT variant is much bigger compared to the direct n-best MHT variant. The semi-log plot visualizes well the growth in the number of generated hypotheses. It is quadratic in the n-best

[2] A *semi-log plot* is a plot where one axis is scaled logarithmically and the other one linearly.

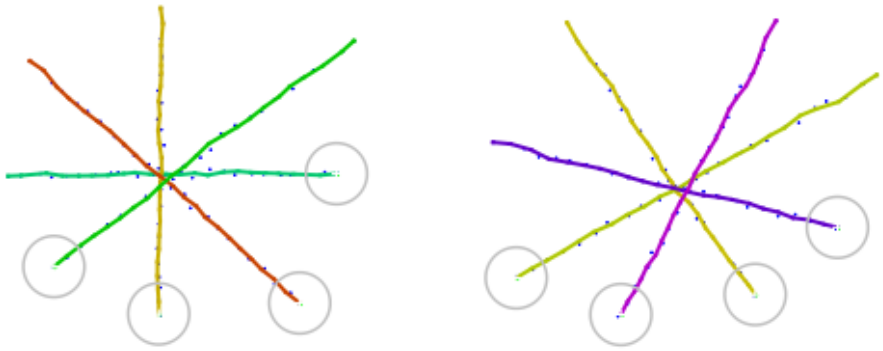
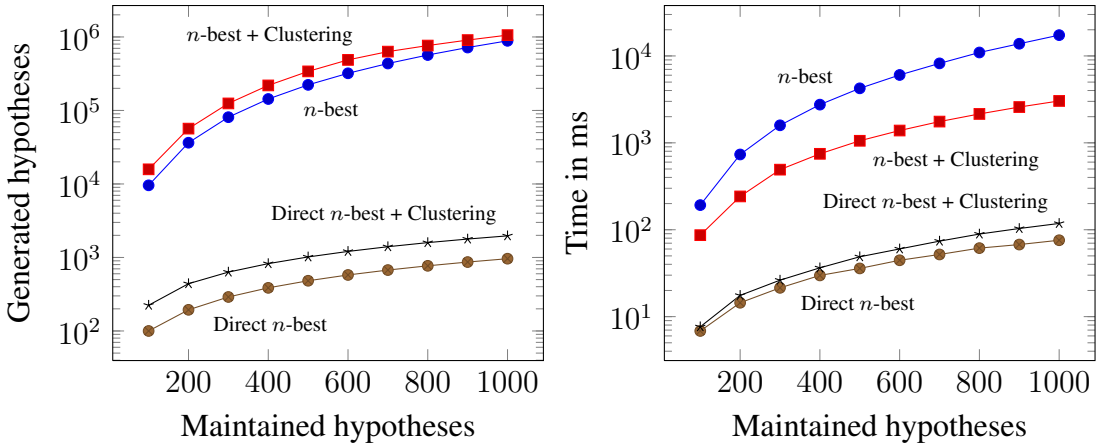


Figure 6.5: Tracking scenario that is used to evaluate the speed of the different MHT variants.



(a) Average number of hypotheses **(b)** Average update time

Figure 6.6: The average number of generated hypotheses (left) and the average update time (right) for processing a scan as a function of the number of maintained hypotheses.

variant and linear in the direct n-best variant. The MHT variants with clustering enabled generate around twice as many hypotheses because for each cluster a set of hypotheses has to be generated. Hence, two clusters result in twice as many hypotheses.

The average time (in milliseconds) required to process one scan evolves as well quadratically for the n-best variant, respective linearly for the direct n-best variant. This is an interesting result because it means that the algorithmic complexity of both n-best variants depends only on the number of hypotheses to maintain, given that the average number of new measurements is constant. By looking at the number of solved LAPs in figure 6.7 and by inspecting Murty's algorithm, it can be be seen that the average number of solved LAPs only depends secondarily on the number of hypotheses to generate but primarily

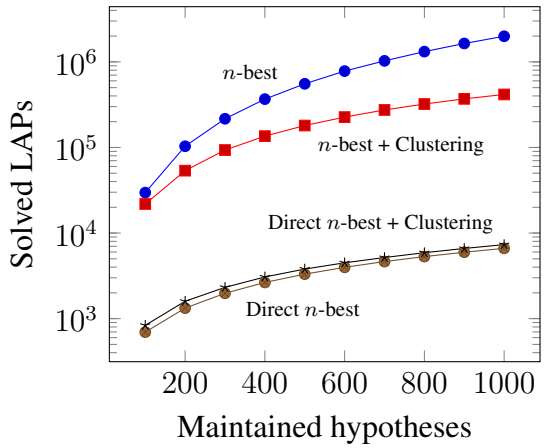


Figure 6.7: Average number of solved LAPs per scan as a function of the number of maintained hypotheses.

on the size of the cost matrix. When a node is partitioned, the number of new nodes and therefore the number of LAPs to solve only depends on the number of columns of the used cost matrix.

This relation also explains why n-best MHT with clustering performs better than without it, even though more hypotheses have to be generated. The reason is that while two clusters are maintained, the number of hypotheses that has to be generated is greater but the cost matrices that have to be solved within each cluster are considerably smaller. With clustering enabled, in the very beginning each cost matrix has only 3 columns ($M_k = 1 = N_{TGT} \Rightarrow L \in \mathbb{R}^{1\times 3}$), later, when the tracks are crossing 12 ($M_k = 4 = N_{TGT} \Rightarrow L \in \mathbb{R}^{4\times 12}$). In comparison, when clustering is disabled each cost matrix has 24 columns ($M_k = 8 = N_{TGT} \Rightarrow L \in \mathbb{R}^{8\times 24}$) when the tracks are crossing. Figure 6.8 illustrates the strong dependence of the update time on the size of the cost matrix.

6.2.1.3 Tracking Quality

In this section the tracking quality of synthetic scenarios is evaluated. For that purpose, two especially difficult scenarios were constructed with the simulation creator. In the first one, targets are moving in close proximity. In the second one, a single target moves in a highly cluttered environment. The CLEAR MOT metrics are used to measure tracking quality.

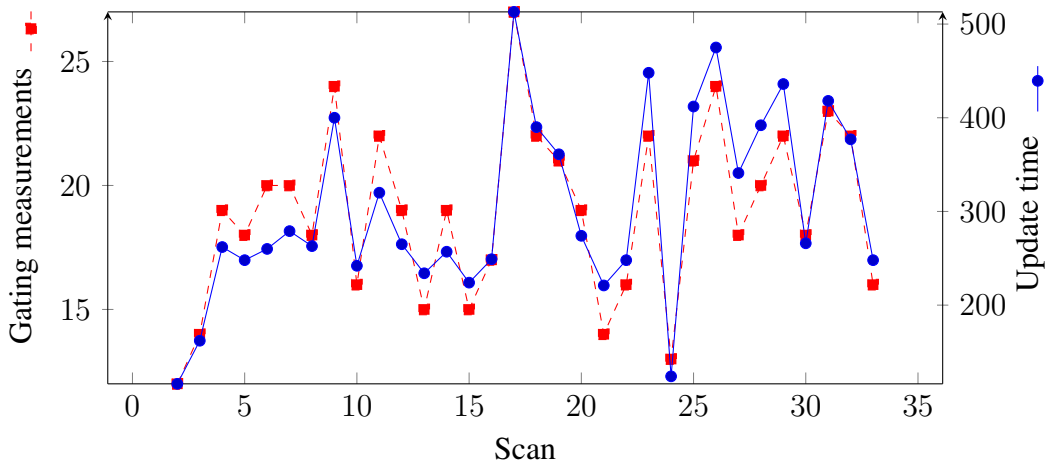


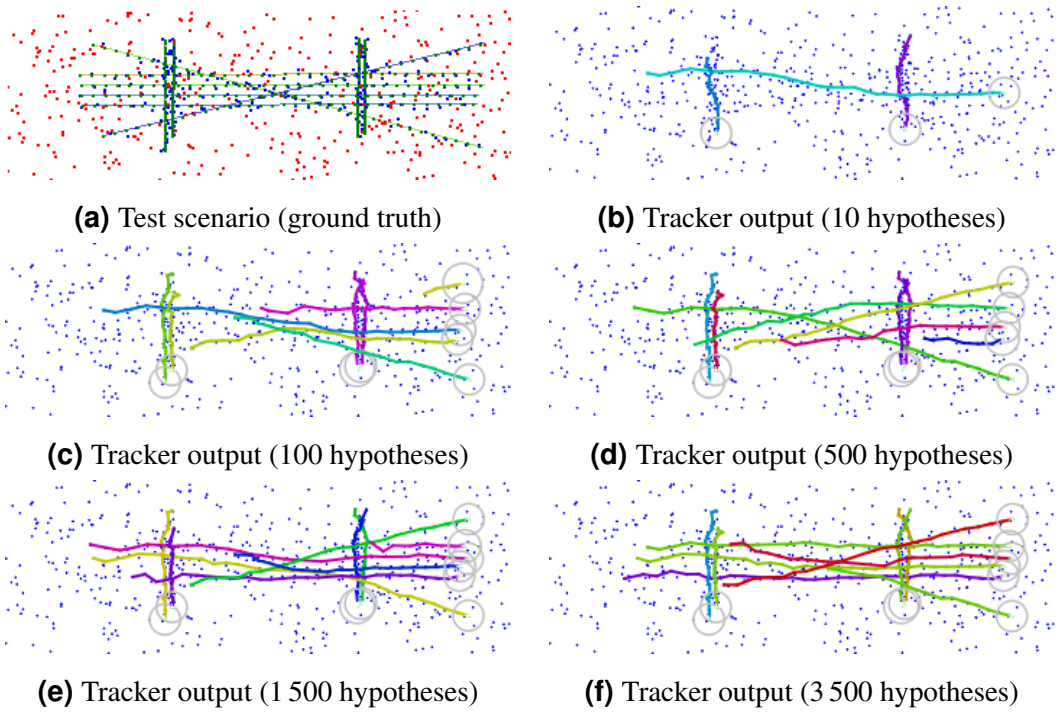
Figure 6.8: This plot shows the strong correlation between the number of gating measurements and the update time.

6.2.1.3.1 High Target Density Scenario Aiming at evaluating the ability of the MHT algorithm to track targets moving in close proximity, a scenario was created consisting of sensor data representing four horizontally, four vertically and two diagonally closely moving targets. The amount of clutter λ is average and the detection probability P_D high. The used tracking parameters are listed in the table below.

Scans	34
P_D	0.85
λ	10
β_{NT}	$\approx 0.000\,007\,002\,8$
β_{FT}	$\approx 0.000\,258\,201\,1$

In figure 6.9a the tracking scenario (blue dots are measurements, red dots are false alarms and lines are targets) and in figure 6.9b– 6.9f the tracker output (blue dots are measurements) for different numbers of maintained hypotheses are shown.

It is clearly visible that as the number of maintained hypotheses is increased the tracking quality improves. The algorithm correctly estimates all of the existing targets and their trajectories only at around 3 500 maintained hypotheses. However, their starting points are still a bit off. Until around 100 maintained hypotheses the number of targets is not detected correctly. With more than 100 maintained hypotheses the number of targets is estimated correctly but their trajectories are still not. Cutting and swapping tracks be-



(a) Test scenario (ground truth)

(b) Tracker output (10 hypotheses)

(c) Tracker output (100 hypotheses)

(d) Tracker output (500 hypotheses)

(e) Tracker output (1 500 hypotheses)

(f) Tracker output (3 500 hypotheses)

Figure 6.9: The high target density test scenario and the tracker output for different numbers of maintained hypotheses.

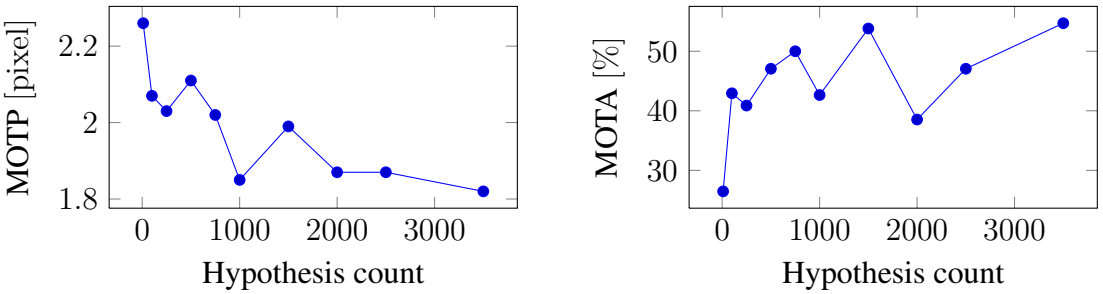
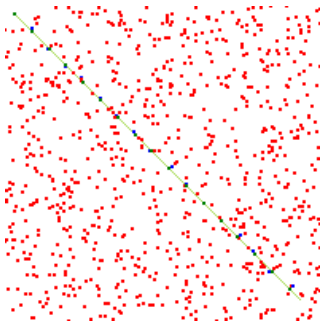


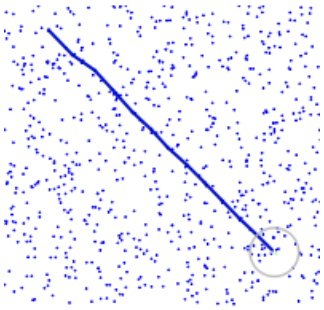
Figure 6.10: The CLEAR MOT metrics as a function of the number of maintained hypotheses.

come the major issues. For instance, in figure 6.9d the upper green track turns left on half of its way. The same happens to the light green track coming from the left. It turns right on half of its way and then merges into the target coming originally from the very upper left.

In figure 6.10 the CLEAR MOT metrics as a function of the number of maintained hypotheses are depicted. It strikes out that neither the tracking precision nor the tracking accuracy graph is *monontonic*. One could assume that as the number of hypotheses is



(a) Test scenario (ground truth)



(b) Tracker output (800 hypotheses)

Scans	16
R	0
P_D	0.6
λ	50
β_{NT}	$\approx 0.000\,001\,471$
β_{FT}	$\approx 0.001\,273\,529$

(c) Used parameters

Figure 6.11: The tracking scenario (left) and the tracker output (right) of tracking a single target in a high clutter environment.

increased the tracking quality improves inevitably. Evidently, this is not the case. When tracking with $m \in \mathbb{N}$ hypotheses it can happen that the hypotheses $\Omega_a, \ldots, \Omega_b$ are generated which eliminate the hypotheses $\Omega_c, \ldots, \Omega_d$ because they are less probable in the current time-step. Though, it is possible that a posterior hypothesis of $\Omega_c, \ldots, \Omega_d$ becomes potentially the most probable one in subsequent time-steps. When tracking with less hypotheses $n < m$ there are cases where the eliminating hypotheses $\Omega_a, \ldots, \Omega_b$ are not created because they are too unprobable in the time-step they would be enumerated.

6.2.1.3.2 High Clutter Scenario

In this scenario a single track in a higly cluttered environment is tracked. In figure 6.11 the tracking scenario is depicted. The target is only detected with a probability of $P_D = 0.5$ and the average number of false alarms per scan is $\lambda = 50$. The remaining tracking parameters are listed in table 6.11c.

At least 800 hypotheses are required in oder to successfully track the single target. The need of such a big number hypotheses is explained by the MHT algorithm requiring multiple scans until the Kalman filter gets more certain and concomitantly the association probabilities for the correct measurements increase. It is crucial to keep at least one hypothesis containing the correct track that started multiple scans before until that point. If not enough hypotheses are maintained all information about the correct track and the prior occurring measurements are lost and the tracker has no information about their existence.

Figure 6.12: Frame 3 444 of the used real-world scenario. The red rectangle is the region where ground truth was extracted for. The blue one depicts the area of coverage of the sensor. The green rectangle is their intersection. Only ground truth and sensor output measurements that fall into this region are considered.

6.2.2 Real-World Scenario

The real-world scenario is a video captured with a surveillance camera installed on the campus of the TU Berlin. The camera is statically mounted and it records a pathway for pedestrians. Therefore, the video contains almost exclusively passing pedestrians with a few exceptions (e. g. cyclists). A *head detector* [DT05] is used for detecting the people present in each video frame. The head detector constitutes the tracking sensor. The locations of the heads of the detected people are the measurements handed over to the MHT algorithm. The last missing piece of information required, before people in the video can be tracked are the parameters P_D, β_{NT} and β_{FT}. A way to determine them will be outlined in the next section.

6.2.2.1 Determining the Tracking Parameters

One problem with using real-world datasets for evaluation is that the detection probability P_D and the densities β_{NT} and β_{FT} are unknown in most of the cases. In order to properly apply any tracking algorithm it is important to estimate these parameters. The

parameters used for evaluating the real-world dataset were estimated using the following approach.

1. For each scan k an assignment problem is solved. The states of the sensor output X^k are assigned to the states of the ground truth G^k by minimizing their overall distance. To avoid unreasonable assignments, a threshold $T_{PE} \in \mathbb{R}^+$ has to be provided which specifies the maximum distance between two states. This works similarly to the threshold used in the CLEAR MOT metrics (see section 6.1.2).

2. The number of detected measurements N_D, the number of false alarms N_C and the total number of tracks N_T which are later used to calculate the final parameters, are updated by using the number of successfully assigned measurements b_k and the number of measurements which got assigned to ground truth states which track labels were encountered for the first time h_k:

$$N_D \leftarrow N_D + b_k \qquad N_C \leftarrow N_C + |X_k| - b_k \qquad N_T \leftarrow N_T + h_k$$

3. Given the counts N_D, N_C, N_T and the scan volume V, the tracking parameters can be finally calculated. The parameters β_{NT} and β_{FT} are divided by V because they constitute spatial densities and not occurrence probabilities.

$$P_D = \frac{N_D}{|G_k|} \qquad \beta_{NT} = \frac{N_T}{|X_k|V} \qquad \beta_{FT} = \frac{N_C}{|X_k|V}$$

6.2.2.2 Tracking Quality

The tracking parameters, required to apply the MHT algorithm on the video, were estimated with the aforementioned method for $T_{PE} = 20$ pixels. The result is shown in the table below.

Number of video frames	10 410
Number of measurements	29 010
Average measurements per scan	≈ 2.787
Detection probability P_D	$\approx 0.548\,776\,284\,0$
New target density β_{NT}	$\approx 0.000\,000\,298\,8$
False target density β_{FT}	$\approx 0.000\,001\,741\,2$

MOTA	$\approx 57.97\%$
MOTP	≈ 4.48 pixel
$\varnothing\ fp$	$\approx 15.96\%$
$\varnothing\ m$	$\approx 24.54\%$
$\varnothing\ mme$	$\approx 1.53\%$

(a) Metrics for GNNT

MOTA	$\approx 55.0\%$
MOTP	≈ 4.68 pixel
$\varnothing\ fp$	$\approx 10.84\%$
$\varnothing\ m$	$\approx 32.54\%$
$\varnothing\ mme$	$\approx 1.62\%$

(b) Metrics for MHT

Table 6.1: The CLEAR MOT metrics of GNNT and MHT for the entire video.

As no reference results for the used real-world dataset exists, it is not possible to make a comparative evaluation. Therefore, a simple GNNT (see section 3.3.1) was implemented to which the MHT algorithm can be compared. The GNNT implementation supports as well track initiation and termination.

The CLEAR MOT metrics of both algorithms are shown in table 6.1. The results are surprising. Both algorithms perform almost equally well. The GNNT algorithm performs even a little bit better (MOTA is 2.97% and MOTP is 0.2 pixel better) than the MHT algorithm. Considering the low algorithmic complexity of GNNT in comparison to MHT the question arises why MHT does not give considerably better result than GNNT.
The graphs in figure 6.13 show the number of false positives, misses and mismatches that occurred within 500 successive time-steps. When looking at the graphs it is visible that the number of false positives, misses and mismatches is similar to a great extent. Hence, the MOTA and the MOTP metrics are similar too because little differences are canceled out as the metrics are calculated as averages over all frames. GNNT performs so well in comparison to MHT for two reasons.

1. There are almost no false alarms in the dataset and thus, the MHT algorithm cannot pay off its big strength of maintaining multiple hypotheses to revoke once wrongly made associations in future time-steps.
2. The GNNT algorithm has a very low rate of misses because it starts tracking any target immediately.

Apparently, a simple nearest neighbor tracker is sufficient for this dataset. This assumption is supported additionally by the dependence of the tracking result on the number of maintained hypotheses. In figure 6.14 the MOTA and MOTP metrics are depicted for different numbers of maintained hypotheses. The CLEAR MOT metrics do not improve

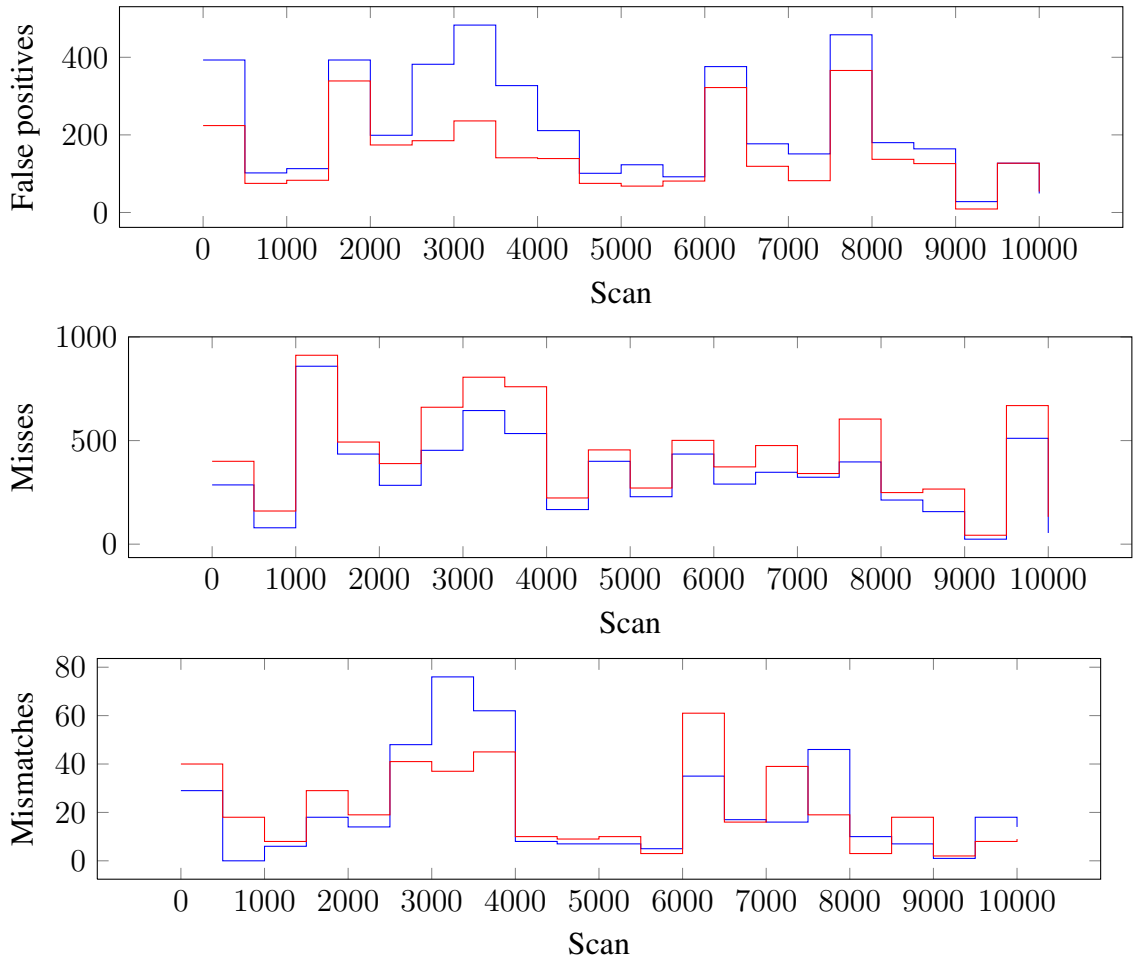


Figure 6.13: The number of false positives, misses and mismatches that occurred within 500 successive time-steps. — graphs belong to the GNNT and — to the MHT.

considerably anymore when more than around 30 hypotheses are used.

It strikes out that the number of false positives and mismatches increases drastically in the frames 3 000–4 000. In these frames the tracking scenario gets complex as a large number of people appear. Comparing the results of MHT and GNNT only for these frames reveals that for complex tracking scenarios the MHT algorithm performs considerably better than GNNT. The results of the CLEAR MOT metrics for the frames 3 200–3 700 are given in table 6.2. One frame of this sequence is shown in figure 6.12.

To study the effect of clutter on the tracking quality synthetic, spatially uniformly distributed clutter was added to the sensor output of the video. In steps of 10 the clutter rate

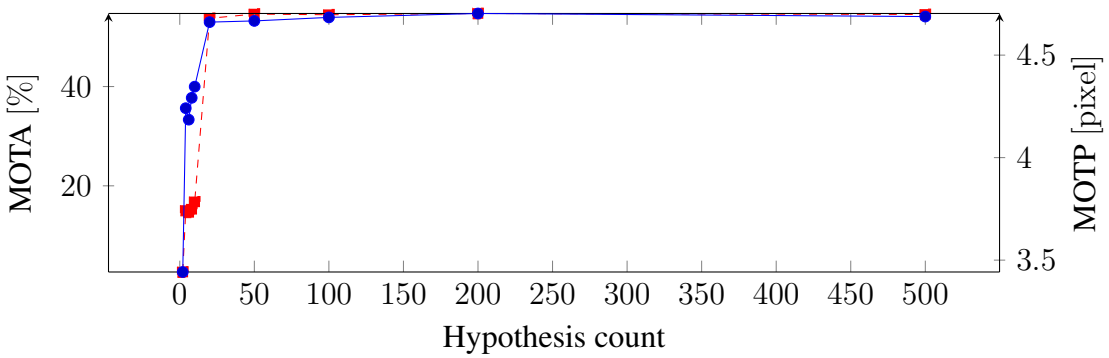


Figure 6.14: The CLEAR MOT metrics as a function of the number of maintained hypotheses for the MHT algorithm. -■- is the MOTA metric, —●— is the MOTP metric. More than 30 hypotheses do not give considerably better tracking results.

MOTA	$\approx 45.71\%$
MOTP	≈ 5.72 pixel
$\varnothing\ fp$	$\approx 26.71\%$
$\varnothing\ m$	$\approx 24.58\%$
$\varnothing\ mme$	$\approx 3.01\%$

(a) Metrics for GNNT

MOTA	$\approx 54.99\%$
MOTP	≈ 5.60 pixel
$\varnothing\ fp$	$\approx 8.59\%$
$\varnothing\ m$	$\approx 35.04\%$
$\varnothing\ mme$	$\approx 1.34\%$

(b) Metrics for MHT

Table 6.2: CLEAR MOT metrics of GNNT and MHT for frames 3 200–3 700. In such complex scenes MHT performs considerably better than GNNT (MOTA is 9.28% and MOTP is 0.12 pixels better).

λ was increased from 0 up to 30 false alarms per scan on average (Poisson distributed). The corresponding CLEAR MOT metrics are shown in figure 6.15.

As the clutter rate λ increases the MOTA metric of the GNNT decreases drastically, whereas the MHT MOTA metric stays approximately constant. The reason for the decreasing GNNT MOTA values are the increasing numbers of false positives and misses. GNNT makes no assumptions about the density of new and false targets. Hence, it outputs a lot of false tracks that contain false alarms as the clutter rate increases. Furthermore, as GNNT does not allow to revert once wrongly made association decisions, tracks cannot be corrected as more information is available.

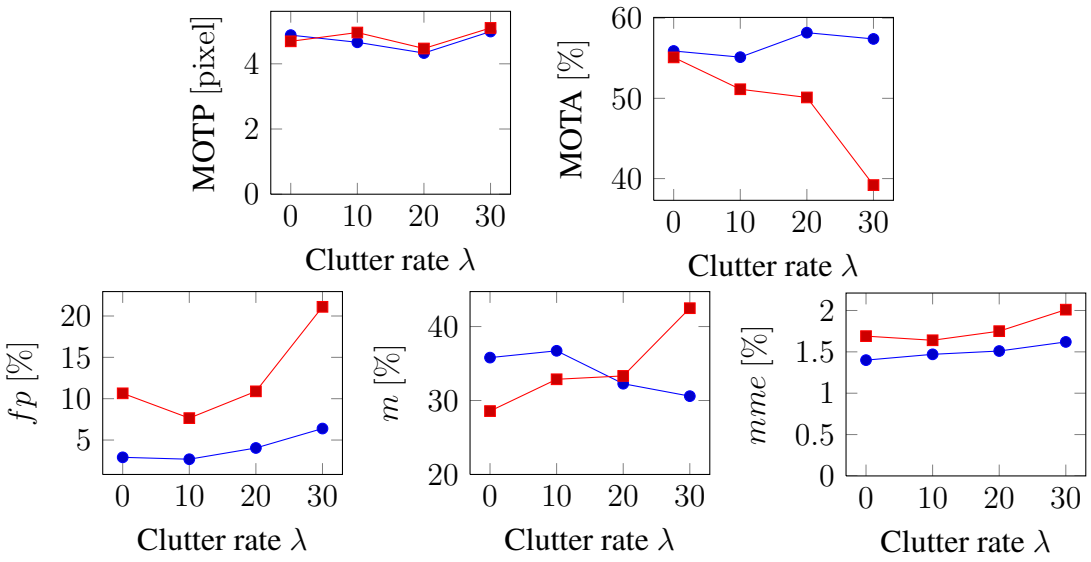


Figure 6.15: The MOTP and MOTA metrics as a function of the clutter rate λ for the real-world scenario. —●— stands for the MHT algorithm, —■— stands for the GNNT algorithm.

6.2.2.3 Summary

The application of the MHT and the GNNT on the real-world dataset revealed that there are situations where simple tracking algorithms are sufficient to achieve results that are comparable to the ones of more complex methods like MHT. Especially, in low clutter scenarios, in scenarios with a high detection probability P_D and in scenarios where the distribution of targets and measurements do not follow the made assumptions of the used algorithm, GNNT performs particularly well. However, in complex scenarios that contain e. g. a lot of crossing targets or a lot of clutter, MHT outperforms any GNNT implementation easily. The ability of MHT to track targets in clutter and close proximity was shown with the use of the synthetic scenarios in section 6.2.1.3.1 and 6.2.1.3.2.

The MHT implementation is fully real-time capable for the purpose of video surveillance. As only around 3 measurements per scan are delivered on average and the number of maintained hypotheses which is required to obtain good tracking results is low, up to 250 scans per second could be processed on the computer that was used for the evaluation. This shows that even a lot more complex tracking scenarios could be handled using the MHT implementation of this thesis.

7 Conclusion

7.1 Summary

In this master thesis the MHT algorithm and a number of optimizations were discussed and implemented. After an introduction to single and multiple target tracking, the conceptional MHT formulation of Reid was discussed in detail. After that, the pruning and the clustering optimizations were presented which speedup the algorithm significantly. However, its inherent exponential character could not be resolved. Therefore, the conceptional MHT algorithm was recast into a linear assignment problem. This took off the exponential complexity and consequently made MHT for the first time applicable to tracking scenarios containing a big number of targets and measurements.

The task of this thesis was to come up with a real-time capable MHT implementation for video tracking in the context of visual surveillance. By analyzing the speed requirements of the implemented algorithm it could be shown that this task was successfully accomplished. The algorithm is fast enough for the simultaneous tracking of many targets in real-time. By adjusting the number of hypotheses the MHT implementation can be adapted to the complexity of the tracking scenario (number of targets and measurements per scan) and almost constant update times can be achieved. Undoubtedly, there is always a trade-off between the tracking quality and the tracking speed.

7.2 Future Work

The field of multiple target tracking is wide and there exist many papers describing further improvements of the MHT algorithm. Some of these papers aim at reducing the execution time of the algorithm, others aim at improving the quality of the tracking results.

Additionally, during the evaluation some issues arose which have potential for further improvement but are not yet treated in literature. Below, the most promising speed and quality optimizations are listed:

Speed Optimizations Optimizing the MHT algorithm for speed is of great importance. The faster the algorithm performs the more targets and measurements can be handled. Moreover, the tracking quality is improved because more hypotheses can be maintained at the same time.

Optimizing Murty's Algorithm Most of the execution time of the n-best MHT variants is spent for solving LAPs in Murty's algorithm. Therefore, optimizing Murty's algorithm further is a major improvement. Miller (1997) [MS$^+$97] describes three optimizations that drastically improve the speed of Murty's algorithm. When used to find the 100 best solutions to a random 100×100 assignment problem, these optimizations result in a speedup of up to 20. Although the average data association matrices in most tracking scenarios are considerably smaller, these optimizations should result in a notable reduction in execution time.

Optimized Gating Procedure As the number of measurements M_k and the number of confirmed tracks N_{TGT} grow, the time required for the gating procedure increases. The issue with the classical gating approach is that its algorithmic complexity is $\mathcal{O}\left(M_k N_{TGT}\right)$. When assuming that for each track, at least, one new measurement is delivered, so $M_k = N_{TGT} + m,\ m \geq 0$, the complexity of the gating procedure is quadratic. Hence, optimizing the gating procedure for speed as described in [CU92] is of interest for tracking scenarios with a large number of targets and measurements.

Parallelization Parallelizing the MHT algorithm can be done in different ways. Parallelizing Murty's algorithm has the advantage that also for single cluster tracking problems the parallelization still brings a speedup. However, parallelizing Murty's algorithm is already a tedious low-level optimization taking place deep inside the MHT algorithm. This means that the amount of parallelized code is small and thus the main thread often *stalls* as it has to wait till all worker threads have finished their jobs. Generally, it is worth to strive for parallelizing as much code as possible as it reduces synchronization costs.

Parallelizing the processing of the individual clusters e. g. enables to run each independent tracking process on a different processing unit without any extra synchronization costs. However, if there are less clusters than processing units some of them are idle. Hence, coming up with a well performing MHT parallelization is of interest.

Hypothesis Management In high clutter tracking scenarios the tracking quality depends almost exclusively on the number of maintained hypotheses. That is why it is of great importance to carefully select which hypotheses are kept and which are discarded to maximize the number of "good" hypotheses that are kept.

Improved Hypotheses Merging Often multiple similar hypotheses emerge that describe almost the same measurement-to-track associations. Such hypotheses can be merged into a single one as described in section 4.3.4.1. This begs the question of how to determine the track states and covariance matrices in the resulting joint hypothesis. A simple solution is to take their averages but there are more sophisticated approaches as, for instance, the one described in [RR10].

Inter Cluster Hypothesis Pruning A single complex tracking problem can be eventually divided into multiple smaller tracking problems by clustering. Each cluster constitutes an independent tracking problem. In the current implementation the same number of hypotheses is maintained in each cluster. Hence, depending on the number of emerging clusters, a different total number of hypotheses has to be generated. This causes the execution time to be highly varying. Hence, a strategy to select a constant number of maintained hypotheses among a variable number of clusters is necessary; especially for hard real-time systems where the update time per scan needs to be constant.

Adapted New and False Target Models In classical tracking algorithms new targets and false alarms are assumed to be uniformly distributed over the scan volume. Furthermore, it is assumed that the measurements are subject to Gaussian noise. In a many tracking scenarios these assumptions do not hold due to the nature of the targets and the sensor. Therefore, it is advisable to come up with improved models that describe better the underlying distributions. For instance, in the paper [BUDW06] it is shown how ellipsoidal gating regions can be extended to the non-linear, non-Gaussian case.

A Fundamentals

In this chapter some important mathematical fundamentals, required throughout this thesis, are presented. An overview of important probability distributions and terms from graph theory is given.

A.1 Probability Distributions

In this section a rough overview of important probability distributions is given. The focus lies on the intuitive understanding of the different distributions and not on providing a mathematical derivation.

Binomial Distribution The *binomial distribution* is a discrete probability distribution [Hay12]. It describes the result of a *Bernoulli process* or *Bernoulli experiment*. A Bernoulli process is a series of $n \in \mathbb{N}$ independent repetitions of a random experiment with only two possible outcomes: success or failure. The outcome A has probability $P(A) = p \in (0,1)$ and the complementary outcome B has probability $P(\bar{A}) = P(B) = 1 - p$. The binomial distribution models the probability to have $k \in \mathbb{N}$, $k \leq n$ successes within n experiments. Its *probability mass function* (PMF) is defined as

$$f_{\mathcal{B}(n,p)}(k) = \binom{n}{k} p^k (1-p)^{n-k}. \tag{A.1}$$

A binomially distributed random variable with parameters n and p is written as $X \sim \mathcal{B}(n,p)$.

Poisson Distribution For large numbers of experiments n and small probabilities p, the binomial distribution is difficult to evaluate [Hay12]. In that case, it can be approximated by the *Poisson distribution*. If the number of expected occurrences

is given by $\lambda \in \mathbb{R}^+$, the *probability density function* (PDF) that there are exactly k occurrences is

$$f_{\mathcal{P}(\lambda)}(x) = \frac{\lambda^k}{k!} e^{-\lambda}. \tag{A.2}$$

A Poisson distributed random variable X with parameter λ is written as $X \sim \mathcal{P}(\lambda)$. A probability distribution $\mathcal{B}(n,p)$ can then be approximated by the Poisson distribution $\mathcal{P}(np)$. Therefore, the number of expected occurrences λ is given by np.

Normal Distribution The continuous *normal* or *Gaussian distribution* is the most important probability distribution as many practical problems are approximately normally distributed [CK07]. A 1-dimensional, continuous, normally distributed random variable X, with mean μ and variance σ^2 has a PDF of

$$f_{\mathcal{N}_1}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right). \tag{A.3}$$

It is written as $X \sim \mathcal{N}(\mu,\sigma)$. A p-dimensional, continuous, normally distributed random variable X, with mean vector $\mu = (\mathrm{E}(X_1), \mathrm{E}(X_2), \ldots, \mathrm{E}(X_p))^T \in \mathbb{R}^p$ and positive-definite covariance[1] matrix $\Sigma = (\mathrm{Cov}(X_i,X_j))_{i,j=1,\ldots,p} \in \mathbb{R}^{p\times p}$ has a PDF of

$$f_{\mathcal{N}_p(\mu,\Sigma)}(x) = \frac{1}{|\Sigma|^{\frac{1}{2}}(2\pi)^{\frac{p}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right). \tag{A.4}$$

It is written as $X \sim \mathcal{N}_p(\mu,\Sigma)$. Analogously to the uni-variate $\left(\frac{x-\mu}{\sigma}\right)^2 = (x-\mu)(\sigma^2)^{-1}(x-\mu)$, the multi-variate case defines the *Mahalanobis distance* [Cox93] as

$$d_M(x,\mu,\Sigma) = (x-\mu)^T\Sigma^{-1}(x-\mu). \tag{A.5}$$

It describes a weighted *Euclidean distance* between the vector x and the mean value μ. By inverting Σ a random variable gets less weight, when it has a greater variance, than another random variable. $(x-\mu)^T\Sigma^{-1}(x-\mu) = c^2$ describes a p-dimensional hyper-ellipse, centered at μ with its shape and rotation defined by Σ.

[1] The *covariance* between two random variables X and Y with mean values $\mu_X = \mathrm{E}(X)$ and $\mu_Y = \mathrm{E}(Y)$ is defined as $\mathrm{Cov}(X,Y) = \mathrm{E}((X-\mu_X)(Y-\mu_Y)) = \mathrm{E}(X,Y) - \mu_x\mu_y$.

χ^2-Distribution The *central* χ^2-distribution is a continuous probability distribution that arises when adding up n independent, squared, p-dimensional, *standard normally distributed*[2] random variables X_i, so that

$$Y = \sum_{i=1}^{n} X_i^2,$$

with $X_i \sim \mathcal{N}_p(0,I)$. It is written as $Y \sim \chi^2(n)$. The parameter n is called the *degree of freedom* (DOF). As the PDF $f_{\chi^2}(x)$ of the central χ^2-distribution is not required in this thesis it is not stated. The CDF, on the other hand, is required and defined as

$$F_{\chi^2(n)}(x) = \frac{\gamma(\frac{n}{2},\frac{x}{2})}{\Gamma(\frac{n}{2})} = P\left(\frac{n}{2},\frac{x}{2}\right), \tag{A.6}$$

where $\gamma(n,z)$ is the *lower incomplete* Gamma function and $P(k,z)$ is the *regularized* Gamma function.

Theorem 3. *The Mahalanobis distance $d_M(x,\mu,\Sigma)$ is χ^2-distributed [DB11].*

Proof. Given the normally distributed random variable $X \sim \mathcal{N}_p(\mu,\Sigma)$, it is

$$(X-\mu)^T\Sigma^{-1}(X-\mu) = (\underbrace{\Sigma^{-\frac{1}{2}}(X-\mu)}_{=:Z})^T(\Sigma^{-\frac{1}{2}}(X-\mu)) = Z^TZ = \sum_{i=1}^{p} Z_i^2.$$

The equation $Z = \Sigma^{-\frac{1}{2}}(X-\mu) = \overbrace{\Sigma^{-\frac{1}{2}}}^{=:A} X + \overbrace{(-\Sigma^{-\frac{1}{2}}\mu)}^{=:b}$ is a linear transformation of the form $Z = AX + b$ of the random variable X with $A \in \mathbb{R}^{p\times p}$ and $b \in \mathbb{R}^p$. Hence, for the mean value and the variance of the transformed random variable Z it holds that

$$\begin{aligned}\mathrm{E}(Z) &= \mathrm{E}(AX+b) = A\,\mathrm{E}(X) + b = A\mu + b = \Sigma^{-\frac{1}{2}}\mu + (-\Sigma^{-\frac{1}{2}}\mu) = 0\\ \mathrm{Var}(Z) &= \mathrm{Var}(AX+b) = A^2\,\mathrm{Var}(X) = (\Sigma^{-\frac{1}{2}})^2\Sigma = I.\end{aligned}$$

Since for the random variables Z_i it holds $Z_i \sim \mathcal{N}_1(0,1)$, the squared sum follows a χ^2-distribution $\sum_{i=1}^{p} Z_i^2 \sim \chi^2(p)$ with p DOF. □

This property is very important because it can be used to determine a threshold $\eta \in \mathbb{R}_+$, so that $P_G \in [0,1]$ of all tested values are less or equal than η:

$$P(d_M(x,\mu,\Sigma) \leq \eta) \leq P_G = F_{\chi^2}(\eta)$$

[2] A *standard normally distributed* random variable has a variance of $\sigma = 1$ and a mean-value of $\mu = 0$.

A.2 Graph Theory Basics

A graph $G = (V,E)$ is called *bipartite* if there exists a partition of its vertices into two subsets $V = X \cup Y$ with $X \cap Y = \emptyset$ and $E \subseteq X \times Y$. In other words: the vertices of G can be partitioned into exactly two sets, where no edges connect two nodes from the same set.

Let G be a bipartite, weighted graph $G = (V,E)$, with $V = X \cup Y$ and weights $w(e) \in \mathbb{R}$, $e \in E$. A *matching* $M \subseteq E$ is a set of edges, where no two edges share a common vertex. The size of a matching $|M|$ is the number of edges in M. The *weight* of a matching is the sum of the weight of its edges $w(M) = \sum_{m \in M} w(m)$. A vertex v is called *matched* if it is an endpoint of an edge in M. If not, v is called *free*. A matching is called *perfect* if all vertices V of G are matched. That means that for any other matching M' of G, $|M'| \leq |M|$ holds [Wal07].

The *neighbor set* of a vertex $v \in V$ is defined as $N(v) = \{v\ (u,v) \in E\}$. The neighbor set of a set of vertices $S \subseteq V$ is defined as $N(S) = \cup_{s \in S} N(s)$ [HR03].

An *alternating path* in G is a path, which edges are alternating between E and $E \setminus M$. An alternating path is called *augmenting* if it holds $v_s,v_e \notin M$ for its start and end vertex. If a matching M contains an augmenting path, then a larger matching M' can be obtained by swapping the edges on the augmenting path [HR03].

A *vertex labeling* is a mapping from the vertices V of G into the real numbers $l : V \mapsto \mathbb{R}$. The associated scalars are called *labels*. A vertex labeling l is called *feasible* if it holds

$$l(x) + l(y) \geq w(x,y),\ \forall x \in X,\ y \in Y.$$

It exists always a feasible vertex labeling by choosing

$$\begin{aligned} l(x) &= \max_{y \in Y} \{w(x,y)\},\ x \in X \text{ and} \\ l(y) &= 0,\ y \in Y. \end{aligned} \tag{A.7}$$

The *equality graph* G_l of G with respect to l is defined as the sub-graph which contains all vertices of G, but only those edges $(x,y) \in E$ that have weights such that $w(x,y) = l(x) + l(y)$ [Wal07].

Bibliography

[Bc98] Rainer E. Burkard and Eranda Çela. Linear assignment problems and extensions, 1998.

[Bc99] Rainer E. Burkard and Eranda Çela. *Handbook of Combinatorial Optimization*, volume Supplement Volume A, chapter Linear Assignment Problems and Extensions, pages 75–149. Kluwer Academic Publishers, 1999.

[BDM09] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[Bla86] S.S. Blackman. *Multiple-target Tracking with Radar Applications*. The Artech House Radar Library. Artech House, 1986.

[Bla04] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):5–18, January 2004.

[Bro89] W. L. Brogan. Algorithm for ranked assignments with applications to multiobject tracking. *IEEE J. of Guidance*, Vol. 12, no. 3:357–364, 1989.

[BS08] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *J. Image Video Process.*, 2008:1:1–1:10, January 2008.

[BSDH09] Yaakov Bar-Shalom, Fred Daum, and Jim Huang. The probabilistic data association filter. *IEEE Control Systems Magazine*, 29(6):82–100, December 2009.

[BSF87] Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and Data Association*, volume 179 of *Mathematics in Science and Engineering*. Academic Press Professional, Inc., San Diego, CA, USA, 1987.

[BSJ72] Y. Bar-Shalom and A.G. Jaffer. Adaptive non-linear filtering for tracking with measurements of uncertain origin. In *Decision and Control, 1972 and 11th Symposium on Adaptive Processes. Proceedings of the 1972 IEEE Conference on*, volume 11, 1972.

[BUDW06] T.A. Bailey, B. Upcroft, and H.F. Durrant-Whyte. Validation gating for non-linear non-gaussian target tracking. *IEEE Conference Proceedings on Information Fusion*, pages 1–6, June 2006.

[BV10] J. Bijsterbosch and A. Volgenant. Solving the rectangular assignment problem and applications. *Annals of Operations Research*, June 2010.

[Can08] Kevin Cannons. A review of visual tracking. *Engineering*, page 242, 2008.

[CH96] I. J. Cox and S. L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(2):138–150, 1996.

[Cha11] Sudha Challa. *Fundamentals of Object Tracking*. Cambridge University Press, Cambridge ; New York :, 2011.

[CK07] E. Cramer and U. Kamps. *Grundlagen der Wahrscheinlichkeitsrechnung und Statistik: ein Skript für Studierende der Informatik, der Ingenieur- und Wirtschaftswissenschaften*. Springer-Lehrbuch. Springer-Verlag, 2007.

[CM95] I.J. Cox and M.L. Miller. On finding ranked assignments with application to multi-target tracking and motion correspondence. *Aerospace and Electronic Systems, IEEE Transactions on*, 31(1):486 –489, jan. 1995.

[Cox93] Ingemar J. Cox. A review of statistical data association for motion correspondence. *Int. J. Comput. Vision*, 10:53–66, February 1993.

[CSD88] J.L. Crowley, P. Stelmaszyk, and C. Discours. Measuring image flow by tracking edge-lines. In *Computer Vision., Second International Conference on*, pages 658 –664, dec 1988.

[CU92] J.B. Collins and J.K. Uhlmann. Efficient gating in data association with multi-variate gaussian distributed states. *Aerospace and Electronic Systems, IEEE Transactions on*, 28(3):909 –916, jul 1992.

[DB11] J.L. Devore and K.N. Berk. *Modern Mathematical Statistics with Applications*. Springer Texts in Statistics. Springer, 2011.

[DF91] Rachid Deriche and Olivier Faugeras. Tracking line segments. *Image Vision Comput.*, 8:261–270, January 1991.

[DF92] O. Drummond and B. Fridling. *Ambiguities in Evaluating Performance of Multiple Target Tracking Algorithms*, pages 326–337. Bellingham, 1992.

[DN93] R. Danchick and G. E. Newnam. A fast method for finding the exact n-best hypotheses for multi-target tracking. *IEEE Transactions on Aerospace*

Electronic Systems, 29:555–560, April 1993.

[DN06] R. Danchick and G. E. Newnam. Reformulating reid's mht method with generalised murty k-best ranked linear assignment algorithm. *Radar, Sonar and Navigation, IEE Proceedings* -, 153(1):13 – 22, feb. 2006.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proc. International Conference on Computer Vision & Pattern Recognition (CVPR2005)*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[dW01] H.W. de Waard. An improved clustering concept for mht applications. In *Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEE*, volume 1, pages 9/1 – 9/8 vol.1, oct. 2001.

[FBSS83] Thomas Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):807–812, 1983.

[FD91] B. E. Fridling and O. E. Drummond. *Performance Evaluation Methods for Multiple Target Tracking algorithms*, volume 1481, page 371. Bellingham, 1991.

[GTK11] A.A. Gorji, R. Tharmarasa, and T. Kirubarajan. Performance measures for multiple target tracking problems. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1 –8, july 2011.

[Har04] Andrew Harvey. *State Space and Unobserved Component Models: Theory and Applications*. Cambridge University Press, 2004.

[Hay12] A.J. Hayter. *Probability and Statistics for Engineers and Scientists*. Cengage Learning, 2012.

[HL01] David L. Hall and James Llinas. *Handbook of Multisensor Data Fusion: Theory and Practice*. CRC Press, June 2001.

[HM02] J.R. Hoffman and R.P.S. Mahler. Multitarget miss distance and its applications. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 1, pages 149 – 155 vol.1, 2002.

[HR03] N. Hartsfield and G. Ringel. *Pearls in Graph Theory: A Comprehensive Introduction*. Dover Books on Mathematics. Dover Publications, 2003.

[HWN08] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pages 788–801, Berlin, Heidelberg, 2008. Springer-Verlag.

[Jaz70] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, April 1970.

[Jun07] Dieter Jungnickel. *Graphs, Networks and Algorithms*. Springer Publishing Company, Incorporated, 3rd edition, 2007.

[JV87] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, November 1987.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[Knu74] Donald E. Knuth. Structured programming with go to statements. *ACM Comput. Surv.*, 6(4):261–301, December 1974.

[Kuh55] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[Kur90] T. Kurien. *Multitarget-Multisensor Tracking: Advanced Applications*, chapter Issues in the Design of Practical Multi-target Tracking Algorithms, pages 43–83. Artech House, 1990.

[KUS03] Pavlina Konstantinova, Alexander Udvarev, and Tzvetan Semerdjiev. A study of a target tracking algorithm using global nearest neighbor approach. In *Proceedings of the 4th International Conference Conference on Computer Systems and Technologies: e-Learning*, CompSysTech '03, pages 290–295, New York, NY, USA, 2003. ACM.

[LSG07] Bastian Leibe, Konrad Schindler, and Luc J. Van Gool. Coupled detection and trajectory estimation for multi-object tracking. In *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, pages 1–8. IEEE, 2007.

[May79] Peter S. Maybeck. *Stochastic Models, Estimation, and Control. Volume I.*, volume 141 of *Mathematics in Science and Engineering*. Academic Press, 1979.

[Mei08] H. Meikle. *Modern Radar Systems*. Artech House Radar Library. Artech House, 2008.

[Mor77] C. Morefield. Application of 0-1 integer programming to multi-target tracking problems. *Automatic Control, IEEE Transactions on*, 22(3):302 – 312, jun 1977.

[MS+97] Matt L. Miller, Harold S. Stone, , Ingemar J. Cox, and Ingemar J. Cox. Optimizing murty's ranked assignment method. *IEEE Transactions on Aerospace and Electronic Systems*, 33:851–862, 1997.

[MS04] D. Meintrup and Stefan Schäffler. *Stochastik: Theorie und Anwendungen*. Statistik und ihre Anwendungen. Springer, 2004.

[Mun57] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[Mur68] Katta G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16(3):682–687, 1968.

[NCS87a] V. Nagarajan, M.R. Chidambara, and R.N. Sharma. New approach to improved detection and tracking performance in track-while-scan radars. part 1: Introduction and review. *Communications, Radar and Signal Processing, IEE Proceedings F*, 134(1):89 –92, february 1987.

[NCS87b] V. Nagarajan, M.R. Chidambara, and R.N. Sharma. New approach to improved detection and tracking performance in track-while-scan radars. part 2: Detection, track initiation and association. *Communications, Radar and Signal Processing, IEE Proceedings F*, 134(1):93 –98, february 1987.

[PHS+06] A. G. Amitha Perera, Anthony Hoogs, Chukka Srinivas, Glen Brooksby, and Wensheng Hu. Evaluation of algorithms for tracking multiple objects in video. *Applied Image Pattern Recognition Workshop,*, 0:35, 2006.

[PPBS99] R.L. Popp, K.R. Pattipati, and Y. Bar-Shalom. Dynamically adaptable m-best 2-d assignment algorithm and multilevel parallelization. *Aerospace and Electronic Systems, IEEE Transactions on*, 35(4):1145 –1160, oct 1999.

[RAG04] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004.

[Rei79] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979.

[RR10] S. Reece and S. Roberts. Generalised covariance union: A unified approach to hypothesis merging in tracking. *Aerospace and Electronic Systems, IEEE Transactions on*, 46(1):207 –221, jan. 2010.

[RVCV11] B. Ristic, Ba-Ngu Vo, D. Clark, and Ba-Tuong Vo. A metric for performance evaluation of multi-target tracking algorithms. *Signal Processing, IEEE Transactions on*, 59(7):3452 –3457, july 2011.

[SB75] P. Smith and G. Buechler. A branching algorithm for discriminating and tracking multiple objects. *Automatic Control, IEEE Transactions on*, 20(1):101 – 104, feb 1975.

[Sch04] H. Schildt. *The Art of C++*. Programming Series. McGraw-Hill/Osborne, 2004.

[SNM06] I. Smal, W.J. Niessen, and E. Meijering. Particle filtering for multiple object tracking in molecular cell biology. In *Nonlinear Statistical Signal Processing Workshop - NSSPW 2006*, pages 44.1–44.4, 2006.

[ST75] Bar Y. Shalom and E. Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica 11*, pages 451–460, 1975.

[Uhl92] J. K. Uhlmann. Algorithms for Multiple-Target Tracking. *American Scientist*, 80:128–141, March 1992.

[Vol96] A. Volgenant. Linear and semi-assignment problems: A core oriented approach. *Comput. Oper. Res.*, 23(10):917–932, October 1996.

[Wal07] W.D. Wallis. *A Beginner's Guide to Graph Theory*. Birkhäuser, 2007.

[WB06] Greg Welch and Gary Bishop. An introduction to the kalman filter. *In Practice*, 7(1):1–16, 2006.

[ZLN08] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

Index