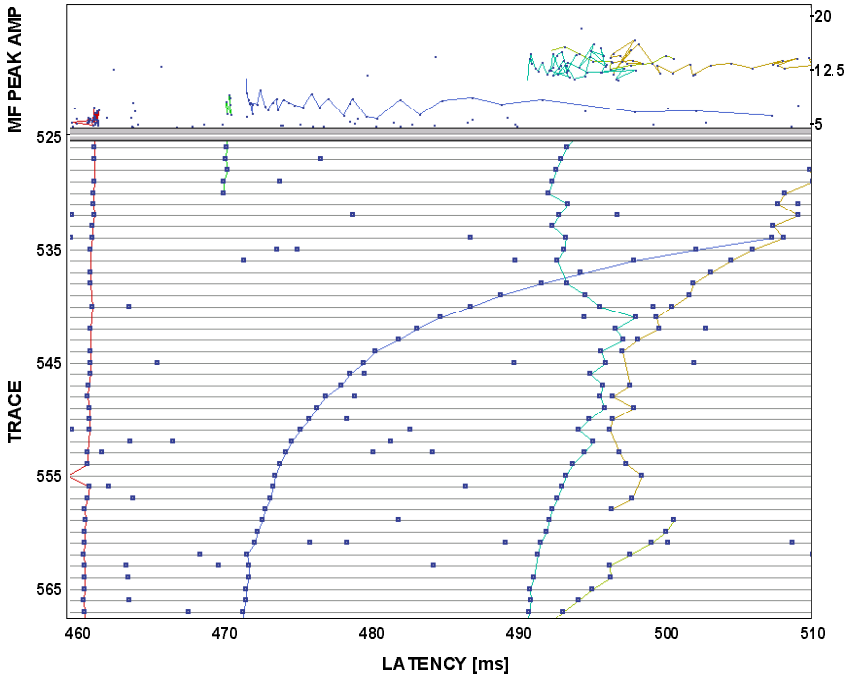
A system for analysis of human pain signals using a radar tracking approach



Author: Karl Lundin

Supervisor: Björn Hansson

Examiner: Mikael Sternad, Signals and Systems Group

Department of Clinical Neurophysiology, Uppsala University, 1998-10-28

Abstract

In order to study peripheral neural activity correlating to pain, an analysis tool that facilitates the investigation of human unmyelinated (C-) fibers has been developed. The tool calculates important C-fiber data such as latency changes and recovery time constants, and helps the researcher to present and statistically process data.

By applying electrical stimuli repetitively at 0.25 Hz and additional stimuli, such as mechanical or chemical, at the receptive field of a studied C-fiber it is possible to estimate important data. The action potentials (APs) that are evoked by the stimuli are recorded. The recordings will usually contain APs from several fibers. Our tool has to determine which C-fiber an AP originated from to be able to get information about the single fibers.

To associate APs to a single C-fiber automatically, an algorithm from the radar tracking area, multiple hypothesis tracking (MHT), is used. The APs are treated as radar targets, and the algorithm tracks and calculates the most probable traces. After tracking, curves are fitted to each trace and C-fiber data are calculated. The analysis program consists of three main parts: detection, association (tracking) of APs and estimation of important constants.

Keywords: Pain, analysis tool, C-fiber, action potential, MHT

1 INTRODUCTION 4

1.1 NEUROPHYSIOLOGICAL BACKGROUND 4

1.1.1 C-fiber 4

1.1.2 Action Potential 5

1.1.3 Recording Method 5

1.2 PURPOSE OF THE MASTER THESIS PROJECT 7

1.3 OUTLINE OF THE REPORT 7

2 AUTOMATIC ALGORITHM OVERVIEW 8

3 MULTIPLE HYPOTHESIS TRACKING – MHT 11

3.1 INTRODUCTION 11

3.2 PREDICTION 12

3.3 GATING 14

3.4 TRACK AND HYPOTHESIS FORMATION 14

3.5 HYPOTHESIS EVALUATION 18

3.5.1 Potential track 19

3.5.2 Tentative track 19

3.5.3 Confirmed track 20

3.5.4 Terminated track 20

3.6 OPTIMIZATION METHODS 21

3.6.1 Shared objects 21

3.6.2 Hypotheses limitation methods 22

3.6.2.1 Combining 22

3.6.2.2 Pruning 23

3.6.2.3 Clustering 23

3.7 IMPLEMENTATION 23

3.7.1 MHT algorithm 23

3.7.2 Classes 24

3.7.3 Data structures 26

4 GRAPHICAL USER INTERFACE 28

5 EXAMPLE 32

6 DISCUSSION 35

7 ACKNOWLEDGEMENTS 36

8 REFERENCES 37

1 Introduction

This report is a master thesis work performed at the Department of Clinical Neurophysiology at Uppsala University. The work is a part of a pain research project in which the Department of Clinical Neurophysiology is collaborating with the Institute of Physiology and Experimental Pathophysiology at the University in Erlangen. The aim of the project is study neurophysiological mechanisms for pain. To get a better understanding of these mechanisms, the nerve signals that carry pain information to the central nervous system are analyzed. Today much of the analysis is performed manually and an automatic analysis tool would make the research more efficient.

The Department of Clinical Neurophysiology has in collaboration with the Signals and Systems Group at Uppsala University, developed an algorithm to automatically analyze nerve signal recordings [1]. The algorithm is based on the manual methods that are used today. The main part of the algorithm is a tracking algorithm, called multiple hypothesis tracking (MHT), developed from the radar tracking area. Why this algorithm is suitable will be clarified later in the report.

This report describes the development of an implementation of the automatic analyzing algorithm. The aim of the system is to facilitate the investigation of pain. The thesis work includes reading the literature in the area and building an application in Microsoft Visual C++. A major effort has been required to understand and implement the MHT algorithm, which is a quite computationally heavy algorithm. The program should have a graphical user interface that makes it easy to carry out the fiber analysis. The graphical user interface should also provide functionality that helps the analysts to present and export data. Below, a brief explanation of the neurophysiological background follows, which is needed to understand the rest of the report.

1.1 Neurophysiological background

A nerve consists of a bundle of many thousands of nerve fibers. Human skin nerves consist of fibers of different types. They are divided into A- and C-fibers depending on their conduction velocity. In this work, the unmyelinated nerve fibers (C-fibers) are considered as they play an important role in pain.

1.1.1 C-fiber

The conduction velocity of the unmyelinated C-fibers is much lower than for class A fibers. This is due to the small fiber diameter (½-1 μm) and the absence of a myelin sheet [2]. The small size of the C-fibers makes them difficult to study and they are quite unexplored. The signals that they transmit are low in amplitude and recordings often contain high levels of noise. The C-unit or C-fiber is sensitive to stimuli in a receptive field at its end. If the receptive field of a fiber is stimulated, action potentials (APs) may be evoked.

C-fibers are divided into afferent and efferent fibers. The afferent fibers conduct signals from the body to the central nervous system. These fibers are divided into subclasses based on their responses

to different kinds of stimuli. The responses are studied by applying e.g. mechanical, thermal, or chemical stimuli. The efferent sympathetic fibers conduct signals from the brain to the body. Sympathetic fibers attend perspiration, for example. The focus in this study is on the afferent fibers.

1.1.2 Action Potential

C-units communicate with the central nervous system by conducting electrical impulses. These impulses are called action potentials (APs) and may be evoked by different kinds of stimuli depending on the sub-class of the C-fiber, see above. Each time an AP arises, it has a constant and maximum strength [2]. After evoking an AP, the conduction velocity of the fiber temporarily decreases. By recording APs of a fiber, it is possible to study some of its properties.

1.1.3 Recording Method

The recordings are performed on healthy young subjects. A microelectrode (0.2 mm diameter) is inserted into the peroneal nerve at knee level to record the action potentials of the C-units. A pair of needle electrodes is inserted into the receptive field (on the foot) and electrical stimuli are applied repetitively, see Figure 1 [3,4]. The electrical stimuli evoke APs that can be detected as spikes in the recording. The amplitude of the APs may be in the same range as the peaks of the noise, and it is a difficult task to identify them.

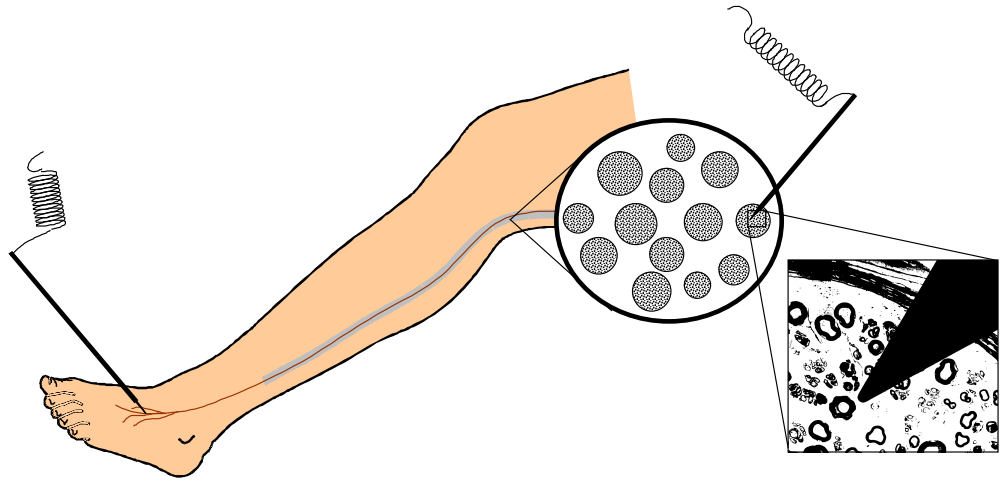


Figure 1: A pair of needle electrodes is inserted into the receptive field of the studied fibers. A microelectrode is inserted into the peroneal nerve at knee level to record the signals. The magnification shows the size of the electrode tip in relation to the diameters of the nerve fibers. (Courtesy of Prof. Erik Torebjörk, Dep. of Clinical Neurophysiology, Uppsala University, Sweden)

The stimuli applied during the experiment usually excite several C-units. If the axons of the C-units are close to the recording electrode tip, the recording will contain APs from different fibers. The amplitude of the APs depends on the distance to the recording electrode. To be able to study single fibers a way to discriminate the recorded APs into different fibers is needed. As mentioned before, C-fibers have low conduction velocity. By using a long distance between the stimulating and the

recording positions, small differences in conduction velocity will result in different latencies and the recorded APs from different C-fibers will be spaced in time, allowing identification of different C-units. Identification of single fibers is facilitated if repetitive electrical stimulation is applied, at a constant frequency. This stimulus will evoke responses (APs) appearing at a stable delay (Figure 2).

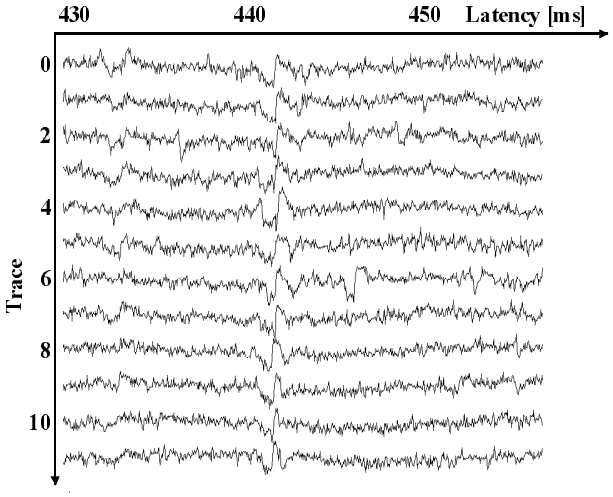


Figure 2: The action potentials from one C-fiber, stimulated with electrical impulses. By displaying successive traces* from top to bottom, the action potentials appear at a characteristic latency. In this recording there is a fiber with latency of approximately 442 ms. The time between successive traces is 4 s (0.25 Hz).

In 1974, Hallin and Torebjörk introduced a method based on what they called the *marking phenomenon* [5]. The method uses the decrease in conduction velocity after conducting an AP to reveal excitation of a specific C-fiber. The marking phenomenon makes it possible to study how single C-units respond to different kinds of stimuli.

The principle of the marking phenomenon is that electrical impulses are applied to the receptive field of the studied C-fiber at a low frequency (0.25 Hz). For each impulse, one single AP is evoked and appears in the recording at a certain latency. To study the response characteristics of a C-fiber, additional stimuli, e.g. mechanical, chemical, or thermal, are applied to the receptive field of the fiber. If such stimuli evoke additional APs, the conduction velocity will decrease. In that case, the AP excited by the electrical stimuli will show a sudden increase in latency and a slow recovery (Figure 3). This change in latency is used as a marker that the C-unit has responded to the applied physiological stimuli.

Figure 3 shows recordings where an additional stimulus has been applied after trace 10. The recording includes two different fibers. The additional stimulus after the tenth trace has evoked extra APs in one of the fibers, and the conduction velocity has decreased. When the additional stimulus is removed, the fiber gradually recovers to the conduction velocity prior to the activation. The magnitude of the latency shift provides an estimate of the number of APs that were evoked by the provocation.

* A trace is the recording of the APs evoked by one electrical impulse

The course of the recovery may be studied and important constants like latency shift and the recovery time constant may be estimated. Recently, it has become evident that the latency shift and the course of the recovery are different for different classes of C-fibers. This finding is interesting since it may be used for a quick identification of different types of C-fibers in a multi-fiber recording.

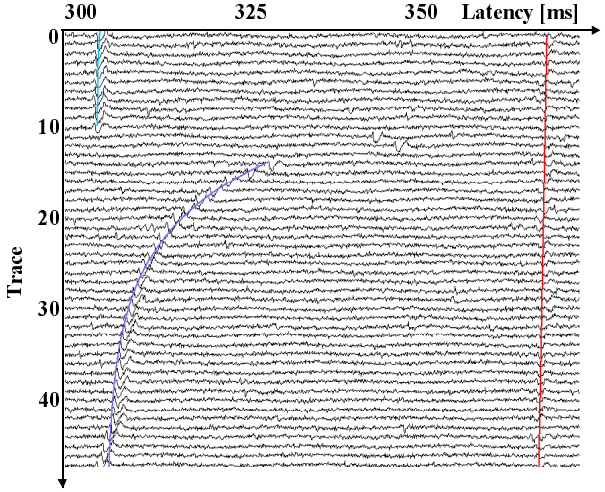


Figure 3: A recording containing APs from two different fibers. Applying additional stimulus after the tenth trace decreases the conduction velocity of one of the fibers (the one to the left). This decrease is used as a marker that the C-unit responded to the applied stimuli. The gradual recovery can be seen in the figure.

1.2 Purpose of the master thesis project

The method described in chapter 1.1 estimates the latencies of individual C-fibers, and detects the latency shifts after activation. Although it is easy to detect APs this way, it is a very tedious process to analyze a recording manually. The goal of this project is to develop an application that automatically detects the APs, measures amplitudes and calculates latency shifts, recovery time constants and other data that may be useful for the analyst.

1.3 Outline of the report

Chapter 2 describes the different parts of the automatic algorithm. The algorithm consists of three main parts: detection, association and parameter estimation. Chapter 3 is the main chapter and it describes the association algorithm, MHT. The theory of the MHT algorithm and the implementation are explained. The graphical user interface is described in chapter 4 followed by a real recording example in chapter 5 to illustrate the performance of the algorithm. Chapter 6 discusses results and possible improvements. The Appendices contains external documentation of the code and the C++ code (on a CD-ROM).

2 Automatic algorithm overview

The three main problems that have to be solved are detection of action potentials, association of APs to different C-units, and estimation of important parameters. The algorithms to solve these problems are explained in [7]. The three main steps are shown in Figure 4.

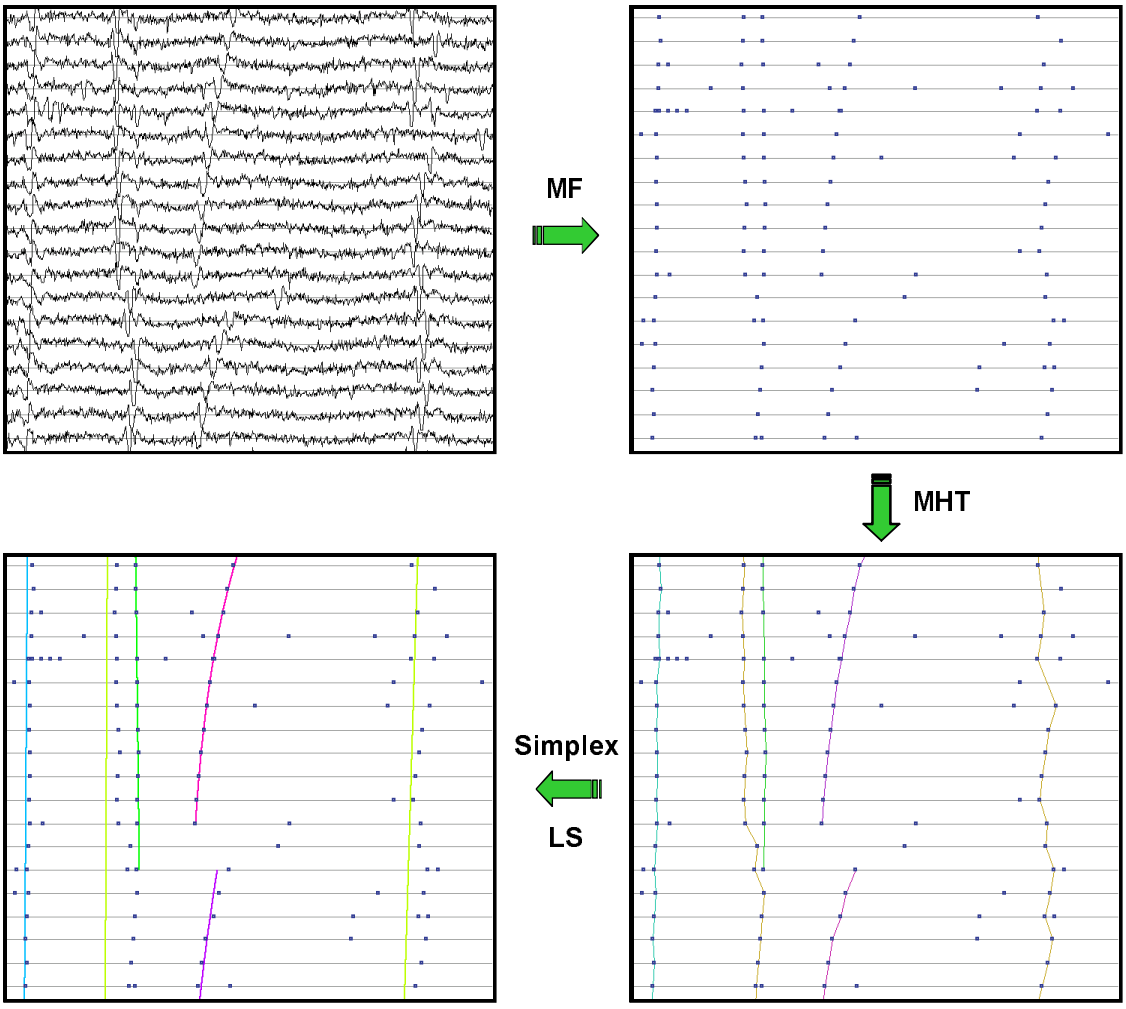


Figure 4: The three main parts of the automatic algorithm. First the AP are detected with a matched filter, then the APs are associated to different paths with the MHT algorithm, and finally a parametric model is fitted to the found paths.

Detection of the Aps, despite high levels of noise, is accomplished by a matched filter. All C-fiber APs are similar in shape and differ mainly by a scalar factor. Under the assumptions that the signal is known and that the noise is wide-sense stationary, the MF is then the optimal detector. These assumptions are not completely true in our application but it has turned out that the MF works fine. The recordings may contain hum from the power lines and, in order to simplify the noise variance estimation, the hum is removed using a notch filter before the actual detection. The filtering is performed using a threshold. Peaks in the filtered signal, with amplitude above this threshold, are reported as APs. The threshold is set at a low value (by the user) and the output from the matched

filter can be thresholded further, using an adjustable threshold. This makes it possible for the user to use a threshold above this value without a time-consuming re-filtering. The MF peak amplitude, the associated latency, and the sample number are stored in the application for further processing.

When the action potentials have been detected, they should be associated to different C-units. This association is complicated and is simplified by associating APs into paths where the analyst manually may associate paths to C-units. We try to mimic the manual procedure by solving the association problem using the multiple hypothesis tracking (MHT) method. The MHT method is a bayesian probabilistic approach to the tracking problem. The implementation of the MHT algorithm is based on [6]. The terminology “target”, “track” and “observation” will be used. In our application, a target corresponds to a path, a track to a C-fiber path and an observation to an AP.

When the APs have been associated to different paths, it is possible to estimate conduction velocity (through the relaxation latency), latency shift and recovery time constant by fitting a theoretical model to the data. The curve fitting is performed using an exponential model. We use the simplex algorithm in combination with the least squares method to estimate the linear and non-linear terms of the exponential model.

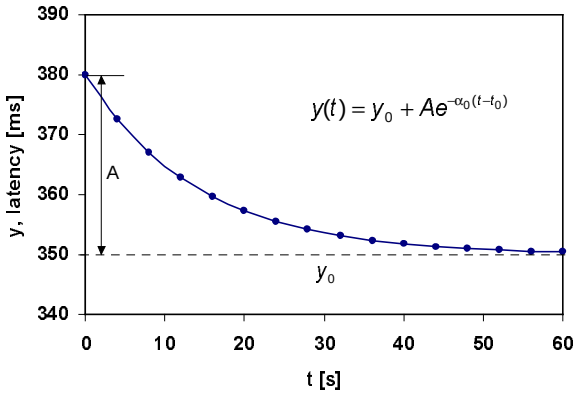


Figure 5: The exponential model used.

The main parts of the application are shown in Figure 6. The document is a container class that stores data and parameters e.g. found APs, found paths, and current threshold. The graphical user interface consists of a number of view classes that display different kinds of information to the user. The data file is the recorded file with some functionality added. The following chapter explains in detail how the MHT object works and how it is implemented.

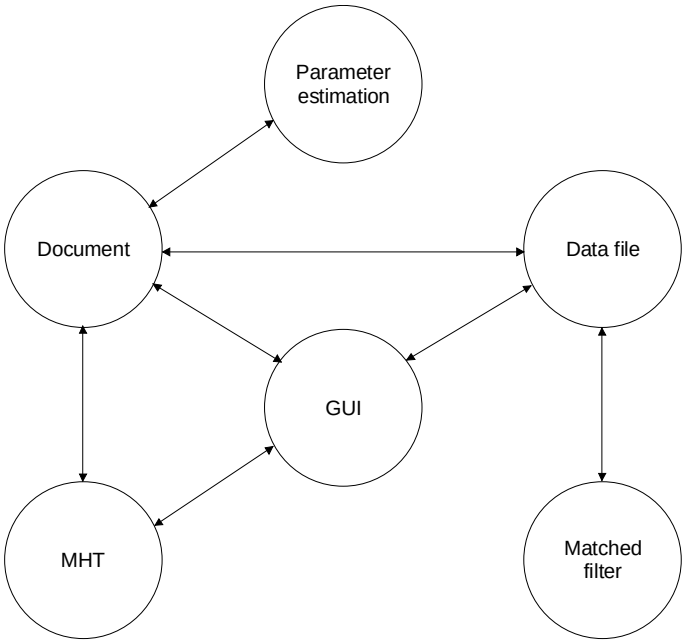


Figure 6: The main parts of the application and how they are associated. The document stores data and parameters. The GUI presents e.g. signals, found APs, found paths, and estimated curves. The data file is the pre-recorded signal file with functionality to find APs added.

3 Multiple Hypothesis Tracking – MHT [6]

3.1 Introduction

Multiple Hypothesis Tracking (MHT) is a tracking method that, unlike simpler tracking algorithms, waits to decide which are the correct associations until further data has been processed. In this application, there will at any given time be a number of plausible ways to combine APs into C-fiber paths. If a standard assignment technique were used, the most likely combination would be chosen after each data set had been processed. This is done, for example, in the nearest neighbor method and may lead to miscorrelation with poor tracking as a consequence.

The basic idea of MHT is to form a number of candidate hypotheses to be further evaluated when more data is available. This approach makes MHT a very good tracking method. Studies show that MHT makes the right decisions in an environment with high probability of false alarm (P_{FA}) at which the nearest neighbor method fails to maintain tracks, see [7]. The MHT algorithm is recursive and each data set is processed only once. A drawback of MHT may be the computational requirements, but in this application there is no need for real-time computing performance.

A path or a track shall be formed from the APs originating from a single fiber. Only APs evoked by the electrical impulses shall be used. A recording may contain some APs from the additional stimuli applied to a certain fiber, but these extra APs shall not be tracked.

The algorithm overview can be seen in Figure 7. Input data are first considered for the update of existing tracks. Gates determine which observation-to-track pairings are "reasonable" so that generation of very unlikely hypotheses can be avoided. When gating is done, the actual track and hypothesis formation begins. To limit the number of hypotheses, some kind of evaluation is needed. This is performed by calculating a score for each track and letting the hypothesis score be the sum of the scores for individual tracks within that hypothesis. Finally, tracks are predicted ahead to the arrival time of the next data set (trace) and the processing cycle repeats itself.

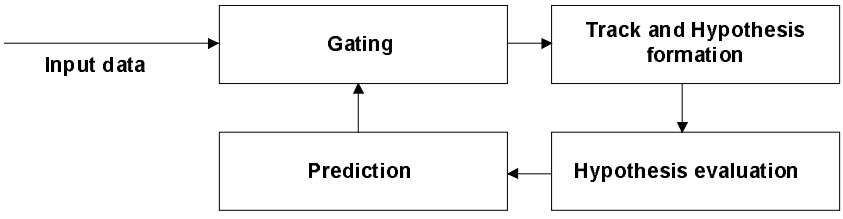


Figure 7: Overview of the MHT logic.

To evaluate the candidate hypotheses, a model of the latency recovery is needed. In most cases, the latency is either constant or approximately exponentially decreasing. Hence, an exponential model is used to describe the time course of the latency. With this model, it is possible to predict future states and evaluate the quality of the different tracks.

The following chapters describe the different steps in Figure 7. Chapter 3.2 describes the prediction logic, which also is used for the gating explained in chapter 3.3. Chapter 3.4 explains track and hypothesis formation, and chapter 3.5 describes the hypothesis evaluation. Chapter 3.6 explains different kinds of optimization methods, and chapter 3.7 focuses on the implementation of the MHT algorithm.

3.2 Prediction

The prediction logic is central in all kinds of tracking systems. Prediction is used to estimate present and future target kinematics such as position, velocity, and acceleration. In this application, the Kalman filter is used due to its simplicity but it is easy to extend the prediction to a more complex method like a filtering method based on interacting multiple models (IMM) [8]. The Kalman filtering generates time-variable tracking coefficients that are determined by an a priori model for the measurement noise and the target dynamics. If the system is linear and the measurement noise is white, with zero-mean, the Kalman filter minimizes the variance of the prediction error. Both the latency and the amplitude information of the APs are used in the prediction logic. For practical reasons, the peak value of the matched filter output is used instead of the real amplitude estimate. The filter is initiated using the first two measurements.

The Kalman filter is derived in [1] but a brief explanation is included in this report because it is needed to understand the gating equations. We assume that the latency follows an exponential curve described by:

$$y(t) = y_0 + Ae^{-\alpha_0 (t-t_0)}, \quad t \geq t_0 \tag{1}$$

where y_0 is the steady state latency, A is the latency shift due to stimulation, α_0 is the recovery time constant and t_0 is the time of excitation. Such an exponential decay can be modeled in discrete time as the impulse response of a dynamic system, described by a second order sampled state space model. This model is combined with a first order random walk model of how the SNR at the matched filter output* changes with time. The total state space model for the states to be estimated by Kalman filtering is then given by:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{v}_1(k), \; k \geq k_0 \tag{2}$$

$$\mathbf{F} = \begin{pmatrix} 1 & T & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad a = e^{-\alpha T} \tag{3}$$

where $\mathbf{x}(k)$ is a three-dimensional state vector consisting of the latency, the derivative of the latency and the square root of the SNR of the matched filter output. Here, α is an a priori guess of α_0 in

* An estimate of the SNR at the matched filter output is needed in other parts of the MHT algorithm.

equation (1), T is the period between traces (constant) and $\mathbf{F}$ is the transition matrix. The three-dimensional vector $\mathbf{v}_1(k)$ is process noise which is introduced to capture errors in the approximate dynamic model. The process noise is modeled as zero-mean, white noise with covariance matrix:

$$\mathbf{Q}_1(k) = \begin{pmatrix} T^3/3 & T^2/2 & 0 \\ T^2/2 & T & 0 \\ 0 & 0 & T \end{pmatrix} \begin{pmatrix} \sigma_v^2(k) & 0 & 0 \\ 0 & \sigma_v^2(k) & 0 \\ 0 & 0 & \rho \end{pmatrix}, \; 0 < \rho << 1 \tag{4}$$

where the possibly time-varying variance function $\sigma_v^2(k)$ describes the modeling error of the latency due to deviation of the constant α from the true value α_o. The recovery time constant is set to 0.06 in current filter. The drift rate ρ for the third state is introduced to describe small changes in SNR at the matched filter output.

The measurement model describes how measurements are collected and is defined as:

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{v}_2(k) \quad \text{where} \tag{5}$$

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6}$$

where $\mathbf{y}\,k$ is the measurement vector containing latency and the matched filter peak output value. The matrix $\mathbf{C}$ maps the state vector to the measurement vector. The two-dimensional vector $\mathbf{v_2}\,k$ is the measurement noise modeled as zero-mean, white noise with constant covariance matrix, $\mathbf{Q_2}$.

Given the target dynamics and measurement models from eq. (2) and eq. (5), the equations for a Kalman filter, which provides filter estimates $\hat{\mathbf{x}}(k \mid k)$ and one step ahead prediction estimates $\hat{\mathbf{x}}(k+1 \mid k)$ of the states in (2), are given by:

$$\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{C}\hat{\mathbf{x}}(k \mid k-1)] \tag{7.1}$$

$$\mathbf{K}(k) = \mathbf{P}(k \mid k-1)\mathbf{C}^T[\mathbf{C}\mathbf{P}(k \mid k-1)\mathbf{C}^T + \mathbf{Q}_2(k)]^{-1} \tag{7.2}$$

$$\mathbf{P}(k \mid k) = [\mathbf{I} - \mathbf{K}(k)\mathbf{C}]\mathbf{P}(k \mid k-1) \tag{7.3}$$

$$\hat{\mathbf{x}}(k+1 \mid k) = \mathbf{F}\hat{\mathbf{x}}(k \mid k) \tag{7.4}$$

$$\mathbf{P}(k+1 \mid k) = \mathbf{F}\mathbf{P}(k \mid k)\mathbf{F}^T + \mathbf{Q}_1(k) \tag{7.5}$$

The vector difference between measured and predicted quantities (latency time and the matched filter peak output value) is defined as:

$$\tilde{\mathbf{y}}(k) = \mathbf{y}(k) - \mathbf{C}\hat{x}(k \mid k-1) \tag{8}$$

with residual covariance matrix:

$$\mathbf{S} = \mathbf{C}\mathbf{P}\mathbf{C}^T + \mathbf{Q}_2 \tag{9}$$

3.3 Gating

Gating is a technique for eliminating unlikely observation-to-track pairings. A gate is an area that is formed around a predicted track position. The gate area has a certain shape and size. An observation is correlated to those tracks that have gates that include the observation. It is common in radar tracking applications to use a maximum likelihood gate where the size of the gate is affected by the prediction uncertainty, see [6]. This gate tended to be too large in our application and a fixed elliptic gate was used instead.

The normalized distance between prediction and measurement is defined as:

$$d^2 = \tilde{\mathbf{y}}^T \mathbf{S}^{-1} \tilde{\mathbf{y}} \tag{10}$$

where $\tilde{y}$ is defined in eq. (8) and $\mathbf{S}$ in eq. (9). Correlation is allowed if the following relationship is satisfied:

$$d^2 \leq G_{fixed} \tag{11}$$

where G_{fixed} is the a priori set fixed gate stored in each track. Because each track stores its gate, it is easy to change to an adaptive gate.

3.4 Track and hypothesis formation

Central in the MHT algorithm is the formation of hypotheses. When a new scan is processed, tracks and hypotheses are generated according to Figure 8. An observation may be correlated with an existing track, with the start of a new track, or with a false alarm. It is assumed that a single target produces no more than one observation per scan.

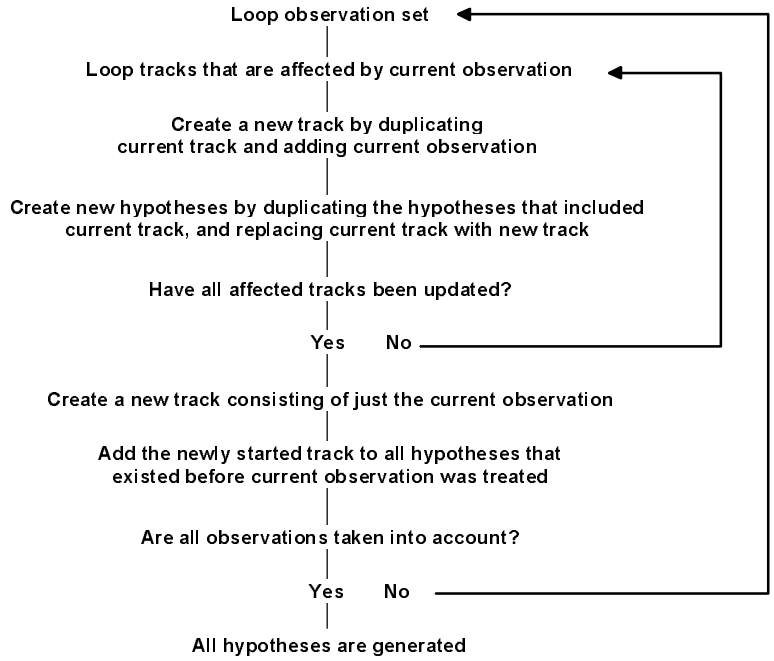


Figure 8: Track and hypothesis formation loop.

The received data set is looped to examine what tracks are affected by the different observations. In order to associate an observation to a track, two important criteria need to be satisfied. First, the observation must lie within the gate of the track. Second, no prior observations must have been previously assigned to that track within the current frame time (scan). These criteria are fulfilled by creating lists of tracks that satisfy the gating equations for each observation in the new observation set. Pointers to the affected tracks are stored in lists that are accessed by the different observations, see Figure 9. This guarantees that a track cannot be created and updated during the same scan.

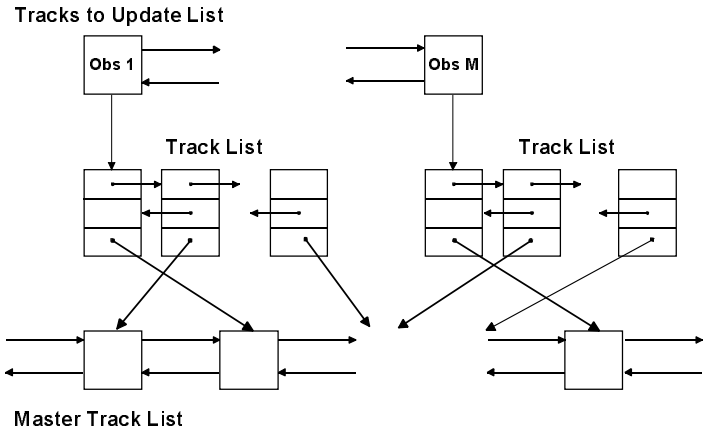


Figure 9: The update list is used to assure that a track is updated only once per scan. The update list stores a list of tracks to update for each observation. Only tracks in these lists may be updated, which guarantees that a track can't be created and updated during the same scan. The master track list data structure is a container for all tracks (this will be explained further in chapter 3.6.1).

Once the creation of this update structure is finished, the formation of new tracks and hypotheses begin. The update structure is traversed so that generation from one observation is completed before the next is considered. For each track in the update structure, a new track is created with observations from the old track plus the new observation. The new track inherits parameters such as gate, detection probability, and Kalman filter from the old track. The new track is added to the master track list and used to create new hypotheses from the hypotheses that included the old track. The new hypotheses are copies from their "parents" but with the old track replaced with the new one. When all tracks that are affected by an observation have been updated, a completely new track, with just the current observation, is created. The new target track is added to all hypotheses that existed before this frame time, to ensure that all hypotheses are current. This formation logic is repeated for all new observations.

This method is illustrated with a small example, where two data sets, with two observations each, is used, see Figure 10. Hypotheses are presented in a matrix form where the rows are hypotheses, the columns are observations, and the elements are tracks. Observations are denoted as $y_j(k)$ (jth observation received on scan k). Tracks are denoted as Ti (ith track). A track created from another track is denoted as $Ti_{new\ track}$ ($Ti_{old\ track}$). A false alarm is denoted as 0.

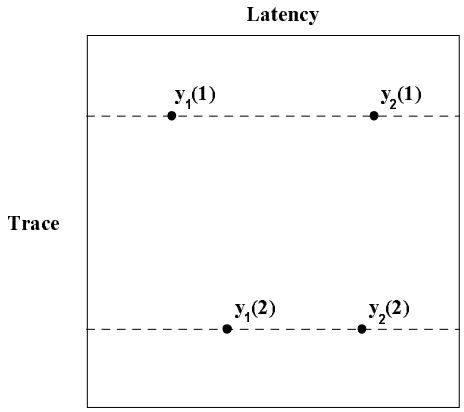


Figure 10: Two data sets with two observations each.

Observation $y_1(1)$ is considered for correlation with existing tracks, but since no tracks existed before this frame, no tracks are affected. The update structure will be empty and only new tracks will be created. Observation $y_1(1)$ may be a false alarm or the start of a new track, *T1*. These alternatives are mutually exclusive and it is possible to create two new hypotheses:

Hypothesis	$y_1(1)$
1	T1
2	0

Table 1: Hypotheses after processing the first observation.

When observation $y_2(1)$ is considered for correlation, there will exist tracks but these tracks have already been updated during this frame time and shall not be updated. Therefore, the possibilities are a new target track, *T2*, or a false alarm. Hypotheses 1 and 2 are replaced by the new ones.

Hypothesis	$y_1(1)$	$y_2(1)$
3(1)	T1	T2
4(2)	0	T2
5(1)	T1	0
6(2)	0	0

Table 2: Hypotheses and tracks after processing the first sweep.

This ends the frame time and the next data set is considered. Now, depending on the gating, the update structure probably will contain tracks. In this example, it is assumed that all tracks are updated with all observations (infinite gate). If we start by considering observation $y_1(2)$, we get the tracks: *T3* from *T1*, *T4* from *T2* and finally a new target track, *T5*. These tracks and the possibility of false alarm expand the old hypotheses list to a total of 12 hypotheses. Hypotheses 3-6 are replaced by the new ones, see Table 3 below.

Hypothesis	$y_1(1)$	$y_2(1)$	$y_1(2)$
7(3)		T2	T3(T1)*
8(5)		0	T3(T1)
9(3)	T1		T4(T2)
10(4)	0		T4(T2)
11(3)	T1	T2	T5
12(4)	0	T2	T5
13(5)	T1	0	T5
14(6)	0	0	T5
15(3)	T1	T2	0
16(4)	0	T2	0
17(5)	T1	0	0
18(6)	0	0	0

Table 3: Hypotheses and tracks after processing one of the observations in sweep two.

In the same way, tracks *T6*, *T7*, and *T8* are created. Updating the affected hypotheses makes a total of 34 hypotheses, see Table 4 below (the parent hypothesis is excluded for convenience).

Hypothesis	$y_1(1)$	$y_2(1)$	$y_1(2)$	$y_2(2)$
19			T4(T2)	T6(T1)
20		T2	T5	T6(T1)
21		0	T5	T6(T1)
22		T2	0	T6(T1)
23		0	0	T6(T1)
24			T3(T1)	T7(T2)
25	T1		T5	T7(T2)
26	0		T5	T7(T2)
27	T1		0	T7(T2)
28	0		0	T7(T2)
29		T2	T3(T1)	T8
30		0	T3(T1)	T8
31	T1		T4(T2)	T8
32	0		T4(T2)	T8
33	T1	T2	T5	T8
34	0	T2	T5	T8
35	T1	0	T5	T8
36	0	0	T5	T8
37	T1	T2	0	T8
38	0	T2	0	T8
39	T1	0	0	T8
40	0	0	0	T8
41		T2	T3(T1)	0
42		0	T3(T1)	0
43	T1		T4(T2)	0
44	0		T4(T2)	0
45	T1	T2	T5	0
46	0	T2	T5	0
47	T1	0	T5	0
48	0	0	T5	0
49	T1	T2	0	0
50	0	T2	0	0
51	T1	0	0	0
52	0	0	0	0

Table 4: Final hypotheses and tracks.

As seen in the last table, the number of resulting hypotheses will be very large even for this small example. If the number of tracks may be decreased, the number of hypotheses will significantly decrease. One easy way to reduce the number of hypotheses is to allow a track with just one

* Track *T3* consists of *T1* plus the new observation and *T1* has been removed from the table.

observation to be either a new target or a false alarm. If this method would have been used in the previous example, the number of resulting hypotheses would have been halved. Another way to decrease the number of tracks and hypotheses is to replace the infinite gate with a smaller one. Additional optimizations methods are discussed in the following chapter.

3.5 Hypothesis evaluation

It is necessary to be able to determine which hypotheses and tracks are the most probable ones. This is done by calculating a score for each track and letting the hypothesis score be the sum of the individual tracks within that hypothesis. The track score function is derived from the maximum-likelihood expression; *-2ln[PR(Z/Q)]*, where *PR(Z/Q)* is the posterior probability of the data *Z* given by the partitioning *Q*. The partitioning *Q* refers to the particular assignment of observation to tracks and false alarms.

Track status is defined in terms of life stages. In this application, we use the commonly defined life stages but with one small difference: confirmed tracks that are deleted are called terminated. C-fibers that respond to additional stimuli make a sudden increase in latency with an ended track as result. Terminated tracks are not considered for updates but are still used for hypotheses evaluation and may be included in the final result. The deletion logic is dependent on the track life stage. The distinction between a single-point hypothesis representing a new true target and a single-point hypothesis representing a false target (i.e. a false alarm) is eliminated for convenience. The score is updated each time an observation is added to a track. After each frame, the score of all tracks that were not updated are calculated recursively. We assume that for each detection (beyond the initial one) associated with a track, the probability density of the residual vector is the Gaussian likelihood function, given by:

$$f(\tilde{y}) = \frac{e^{-\frac{d^2}{2}}}{(2\pi)^{M/2}\sqrt{|\mathbf{S}|}} \tag{12}$$

where d and $\mathbf{S}$ are defined in equation (10). This probability density is converted to a finite probability within space cells (see Figure 11). The log likelihood expression for data association becomes, see [6] chapter 9:

$$L_i = \ln\left(\frac{\beta_{NT}}{\beta_{FT}}\right) + \sum \ln\left[\frac{P_D}{(1-P_D)\hat{a}(2\delta)^{M/2}\sqrt{|\mathbf{S}|}}\right] \tag{13}$$

where β_{NT} is the new target density, β_{FT} is the false target density and M is the measurement dimension. The initial probability of detection is set by the user (the default value is 0.98) and used to calculate the P_D connected to each individual track with more than one observation, see equation (17). The target densities, β_{NT} and β_{FT}, are defined as:

$$\beta_{NT} = \frac{P_{NT}}{V_{cell}} = \frac{I_{NT}}{\Delta Amp} \tag{14}$$

$$\beta_{FT} = \frac{P_{FA}}{V_{cell}} + \frac{I_{clutter}}{\Delta Amp} \tag{15}$$

where I_{NT} is the new target intensity per space cell and $I_{clutter}$ is the clutter intensity [1/ms]. Both these intensities may be changed in the application. V_{cell} is defined to be a measurement volume element such that independent true target detection and false alarm events occur within each element (in this case an area, $\Delta Amp * \Delta Lat$), see Figure 11.

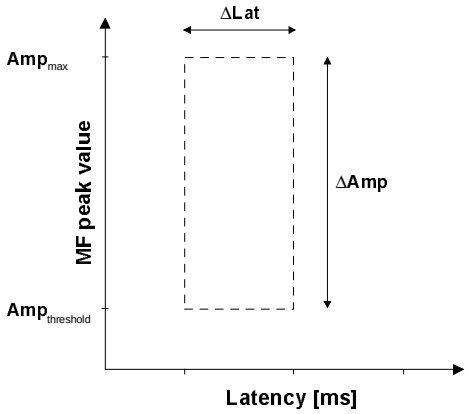


Figure 11: The volume element (space cell) such that independent true target detection and false alarm events occur within each element. The latency width is determined by the resolution of the matched filter. The fine threshold is set in the application and the maximum amplitude is calculated during the matched filtering.

3.5.1 Potential track

A potential track consists of a single observation, which may be the start of a new track or a false alarm. A potential track is deleted if its score falls below a given threshold or if it misses an update opportunity. The score of a potential track is calculated according to:

$$L_i = \begin{cases} \ln\left[1 + \frac{\beta_{NT}}{\beta_{FT}}\right] & i = 0 \\ L_{i-1} + (1 - \hat{P}_D) & i > 0 \end{cases} \tag{16}$$

where $\hat{P}_D$ is the estimated probability of detection (the initial user set value is used) and i is the number of scans since the track was created.

3.5.2 Tentative track

If a second observation is added to a potential track, the false alarm option is eliminated. Like the potential track, a tentative track is deleted if its score falls below a given threshold. A tentative track is

allowed to miss a number of consecutive update opportunities before it is deleted. The detection probability is calculated recursively according to:

$$P_D(k) = (1 - \lambda_{PD})P_D(k-1) + \lambda_{PD}\left[1 - \sigma(a_{thresh} - \tilde{a}(k))\right] \tag{17}$$

where λ_{PD} is the forgetting factor, a_{thresh} is the chosen fine threshold and $\tilde{a}$ is the amplitude estimate. The score for tentative (and confirmed) tracks function is implemented according to equation (13). We have a two dimensional measurement vector (M=2) and the score becomes:

$$L_i = \begin{cases} \ln\left(\dfrac{\hat{a}_{NT}}{\hat{a}_{FT}}\right) & \text{i = 0} \\ L_{i-1} + \ln(1 - P_D) & \text{i > 0, missed detection} \\ L_{i-1} + \ln\left[\dfrac{P_D}{\hat{a}_{FT}\, 2\pi\sqrt{|\mathbf{S}_{il}|}}\right] - \dfrac{d_{il}^2}{2} & \text{i > 0, detection added} \end{cases} \tag{18}$$

where $\mathbf{S}_{il}$ is the residual covariance matrix associated with track i at the lth observation. d^2_{il} is the normalized distance associated with the lth observation of the ith track. The first term (i=0) is the score resulting from the track initiation. The second term is a penalty term arising when detections are not received. This term approaches negative infinity as P_D approaches unity. The final term is the score resulting from an added observation. The magnitude of the final term depends on the distance between the prediction and the added observation, see equation (10).

3.5.3 Confirmed track

The only difference between a tentative track and a confirmed track is in the deletion logic used. A confirmed track must not be deleted and is instead terminated. The termination is performed when the score would be lower than the maximum score achieved, even if a new observation would be added. The score is computed in the same manner as for tentative tracks. A tentative track becomes confirmed if it satisfies the criterion:

$$L_i > L_{conf} \tag{19}$$

where L_{conf} is the a priori set confirmation threshold chosen by the user.

3.5.4 Terminated track

A terminated track is defined as a track that is no longer active but still affects the hypotheses. A terminated track is formed when a confirmed track is deleted. The terminated track is not a candidate for further updates but may be part of the final result. The score of a confirmed track will rise to a maximum and then, if the track is lost, start to decay. The amount of decay relative to the maximum score determines whether termination is to be performed. The termination condition is defined as:

$$L_i < L_{i\max} - \Delta L \quad \text{where} \tag{20}$$

$$\Delta L = \ln\left[\frac{P_D}{(\hat{a}_{NT} + \hat{a}_{FT})2\delta\sqrt{|\mathbf{S}_{ii}|}}\right] - \frac{M}{2} \tag{21}$$

The score of a confirmed track that is ended will be below the maximum score of the track because of the penalty score for the missed update opportunities. When the track is terminated, it should return to its state before the consecutive misses. This means that the score of the terminated track is to be set to the maximum score of the confirmed track.

3.6 Optimization methods

3.6.1 Shared objects

To make an efficient implementation, it is important to notice which objects are shared. The observations may be used by several tracks in different hypotheses. Thus, they are stored in a shared data structure where the tracks use pointers to build lists of correlated observations. The tracks are also shared, because different hypotheses may use the same track. This fact is utilized by storing the tracks in a common data structure in each cluster, called a master track list. Then, each hypothesis stores pointers to the tracks in the master track list.

Track sharing produces some additional complications. It is not possible to delete all tracks used by a hypothesis that is deleted. A mechanism is required in order to decide if a track may be deleted and unused memory to be reclaimed. This is done by counting the number of hypotheses attached to each track and storing this information in the individual tracks. The track sharing implies that there may exist many more hypotheses than tracks. To avoid a track being processed several times, some operations (e.g. gating, calculation of score) are performed directly on the tracks. Often the operations will be performed on all the tracks and it is convenient to add such functionality to the master track list. In the same manner, it is convenient to perform some operations on all hypotheses (e.g. pruning, combining and calculation of scores) and such functionality is added to the hypothesis list.

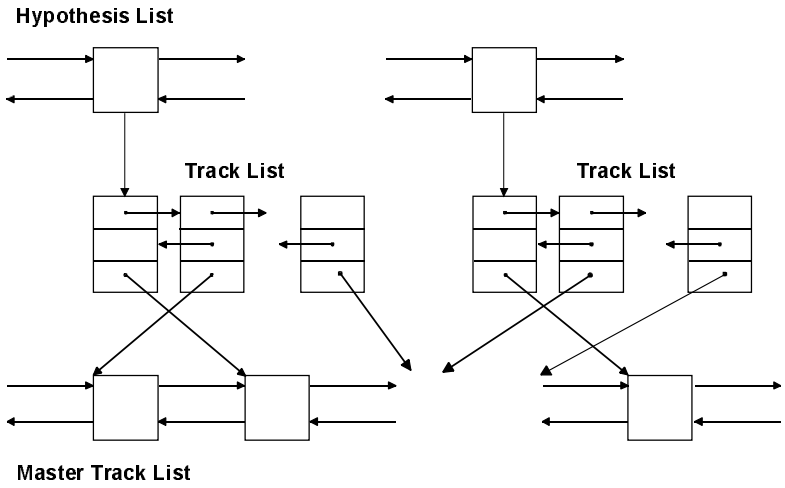


Figure 12: The structure of the hypothesis list. Each hypothesis includes an individual track list. This list is internally double linked and stores pointers to the tracks in the master track list.

3.6.2 Hypotheses limitation methods

3.6.2.1 Combining

Hypothesis limitation methods are required to keep the number of hypotheses at a reasonable level. A way to limit the number of tracks is to combine similar tracks. Combining is done using the N-scan criterion because of its ease of implementation. The N-scan criterion combines tracks that share the N most recent observations. The track with the highest score is chosen to replace the others in each of the hypotheses. The hypothesis score must remain unchanged after track combining because this combining procedure is essentially a way to limit the number of tracks to be stored. This constraint is maintained by storing the score differences in a residual score, SCRES, in each hypothesis. Whenever a track is replaced, SCRES is incremented by the difference in score between the former track and the track that is replacing it. The hypothesis score is calculated as the sum of the score of the tracks contained plus SCRES (negative).

This method works fine if, like in common radar tracking applications, the interest is in the current state. The drawback of the N-scan criteria is that it does not guarantee hypotheses with track sets that are compatible[*] beyond the N latest scans. Tracks in a hypothesis may be replaced by better ones and there is no guarantee that these tracks are compatible with the rest of the track set. This means that tracks may overlap. In the default configuration there is no track combining used but it is possible to turn it on by setting a N-value in the application. To avoid lots of overlapping tracks the N-value should be rather large (>2).

[*] Compatible means that tracks in a set doesn't share observations

Hypotheses that have the same sets of tracks may also be combined. Hypotheses are changed when tracks are combined and when tracks are deleted. In this application, combining hypotheses means dropping the one with the lowest score.

3.6.2.2 Pruning

The most important way to limit the number of hypotheses is through pruning. We use a simple method where the number of hypotheses is limited to the M most likely. The number of hypotheses to be kept after processing each observation and after processing all observations in a frame (scan) is M_1 and M_2, respectively. A larger number of hypotheses should be kept after processing each observation and therefore M_1 should exceed M_2.

The pruning is implemented in a simple way. If the number of hypotheses exceeds $2*M_x$, we select and retain the M best hypotheses. If the number of hypotheses is below $2*M_x$, we delete the worst hypothesis until the limit (M_x, depending on whether the pruning is after an observation or after the hole frame) is reached.

3.6.2.3 Clustering

If the entire set of hypotheses is divided into sets of independent clusters it is possible to increase the calculation speed. A cluster is by definition a set of hypotheses that share a set of observations. Different clusters must not contain the same observation. Validating to which cluster an observation belongs to is done in the gating procedure. If more than one cluster is affected by an observation, the clusters must be merged to form a supercluster. The merging is done by saving one of the involved clusters and create all combinations of the hypotheses that were included in all merged clusters. All tracks, from the merged cluster set, are copied to the supercluster. Once having assigned an observation to a certain cluster, hypothesis generation proceeds only internally to that cluster.

3.7 Implementation

3.7.1 MHT algorithm

The MHT logic used in this application is illustrated in Figure 13. Observations are gated with the existing tracks and an update structure is formed. If observations satisfy gates of tracks within more than one cluster, the clusters are merged. When all observations have been processed and the update structure is completed, the formation of tracks and hypotheses begins. The number of hypotheses is limited by combining and pruning methods. After traversing the whole update structure, track deletion and prediction for the next scan occur. Before the cycle repeats, the best hypothesis from each cluster is sent to the rest of the application and plotted in the main graph.

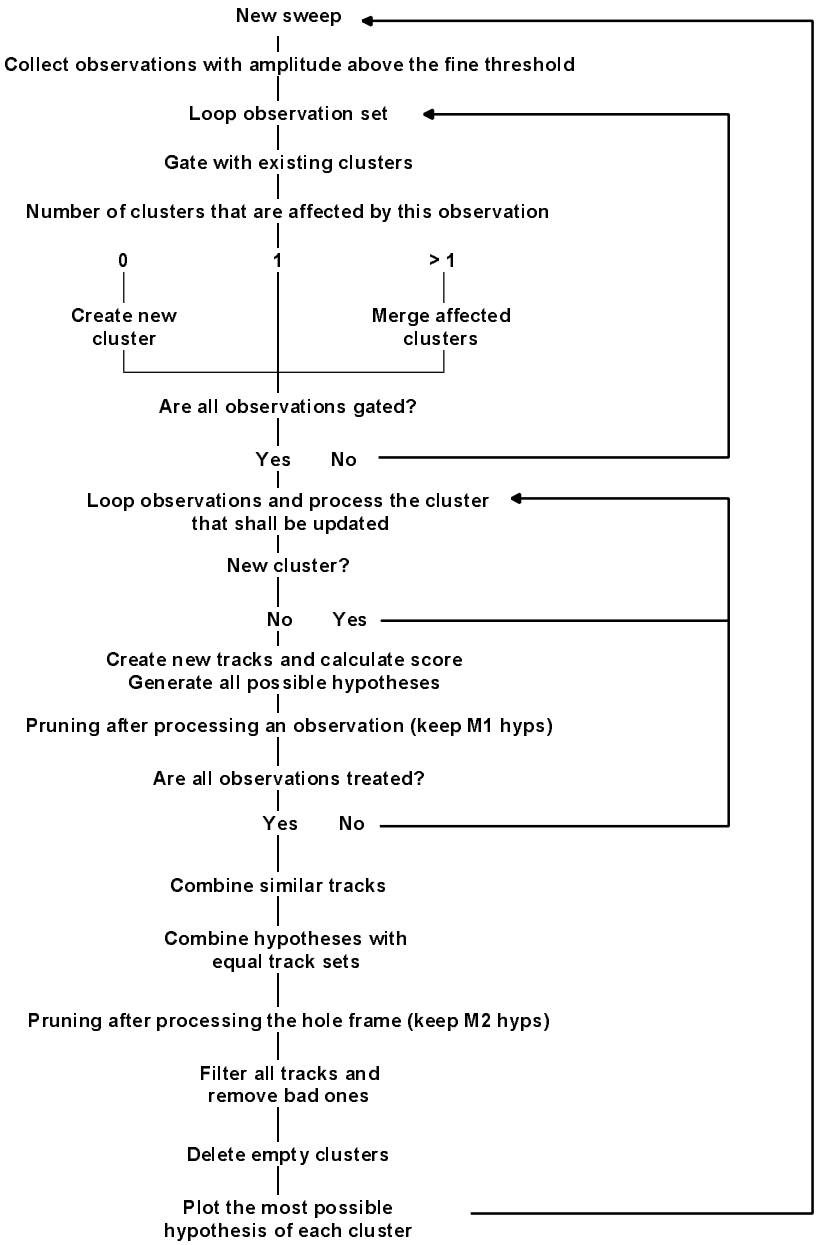


Figure 13: Flow chart of the MHT logic.

3.7.2 Classes

An overview of the multiple hypotheses tracking algorithm implementation is as follows: The MHT object contains a linked list of clusters. Clusters contain linked lists of hypotheses. Hypotheses are represented by some parameters and a linked list of used tracks. Finally, the tracks are represented by some parameters and a linked list of observations. The class objects chosen are the underlined. These are the main MHT classes but some more have been discussed in the text. It has been mentioned that it is convenient to perform operations on whole set of tracks, hypotheses, and clusters. Therefore some additional classes like track list, hypothesis list, and cluster list is added.

The lowest level element in the data structure is the observation object, containing information of a single observation. This class of object contain information about id, sweep number, amplitude, latency, and sample number. It may seem redundant to store both the latency and the sample number but this has turned out to be convenient.

The next level is the track object. This object contains a linked list of observations and some parameters. The parameters include current gate, score, the estimate of the detection probability, a pointer to the Kalman filter (makes it easy to change the prediction technique), and the track life stage. A track actually contains a linked list of hypotheses that are attached to it, because we thought that this would be good. For the moment, this list is unused but remains in the code. This list class is called hypothesis list and the main hypotheses class is called master hypothesis list.

A hypothesis contains a doubly linked list of track pointers, current score achieved from tracks, and a residual score that originate from the track combining.

A cluster contains a master track list, a master hypothesis list, and an update structure used to assure that new tracks aren't updated during the frame time when they where created. A cluster also knows whether it was created during this frame time.

The MHT object contains a list of clusters. It is convenient to perform commands on the entire set of clusters, and thus functionality is added to the cluster list. The MHT class also contains a structure to mark which cluster that shall be updated by each observation. The class overview can be seen in Figure 14.

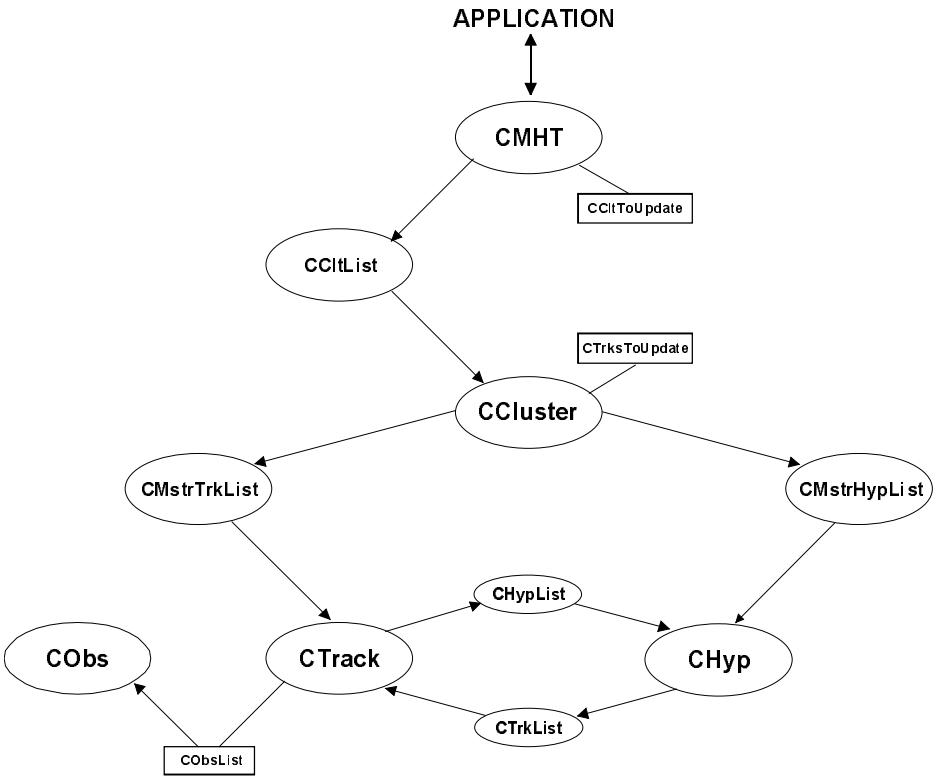


Figure 14: Overview of the MHT classes.

3.7.3 Data structures

All lists are double linked which makes them easy to traverse. Element insertion is fast at the list head and at the list tail. A sequential search is needed to look up an element by value or index. This search method is slow and should be avoided if possible. Most of the lists have functionality added which makes it possible to perform actions on whole sets of objects. An overview of the data structure is presented in Figure 15.

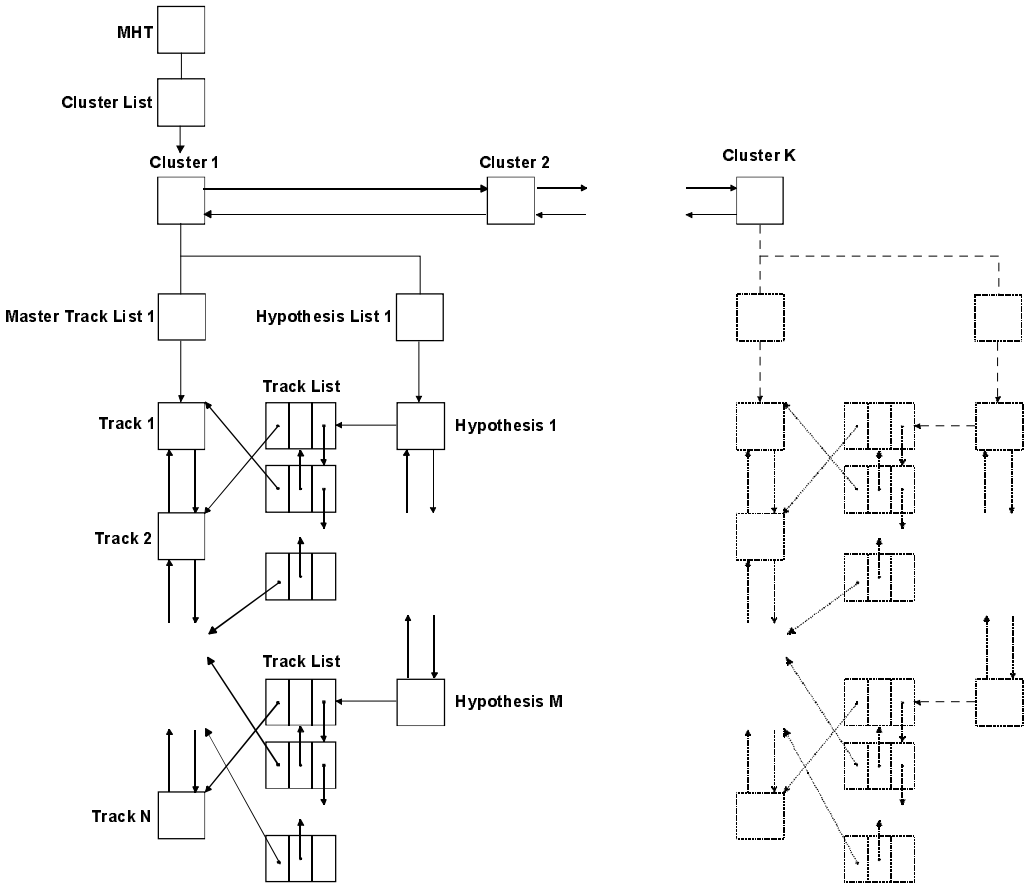


Figure 15: Overview of the used data structures.

The MHT object contains a list of all the clusters. A cluster contains a master track list and a master hypothesis list. The structure of these lists is seen in Figure 15. The track data structure is displayed in Figure 16.

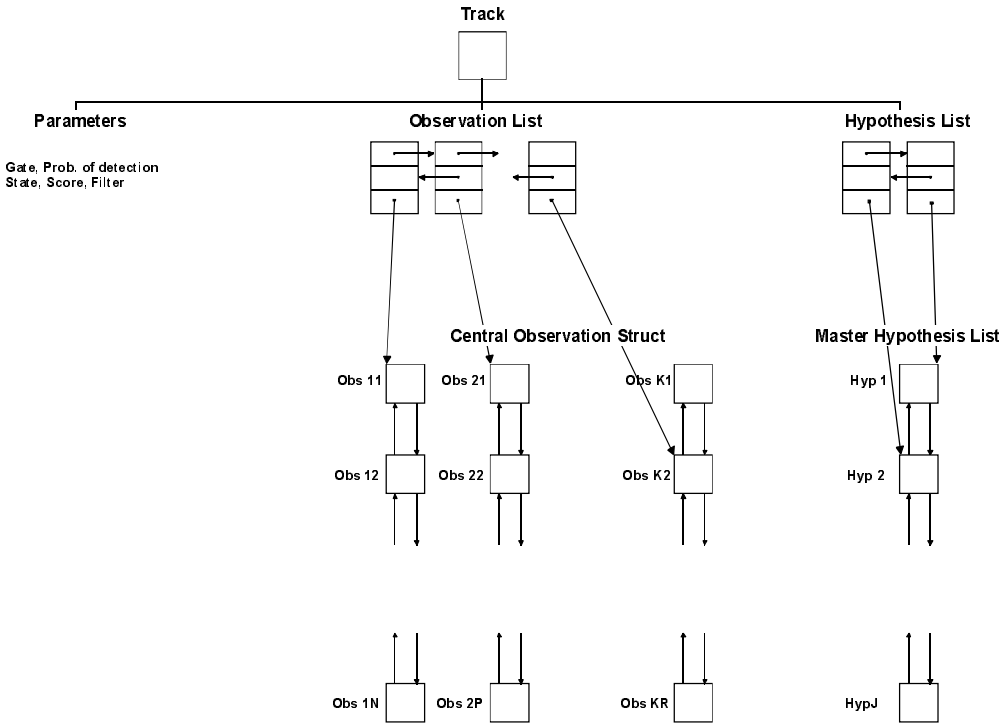


Figure 16: Track object overview.

4 Graphical User Interface

The analyzing tool will be used by experienced analysts. They want to be able to:

- Display information of the chosen file
- Display successive recorded signals
- Display detected APs
- Analyze parts of the data file
- Display the amplitude of the APs and make it possible to set a threshold
- Display found paths
- Calculate important C-fiber data
- Edit paths that are found by the MHT algorithm
- Make presentation and further analysis easy

The GUI is implemented in a way that makes it possible to focus on the parts that is needed for the moment. All windows may be resized and be floating or docked*. Even docked windows may be resized although this affects the other windows. Windows that are not being used for the moment may be closed down and redisplayed when the need arises. The window customization is a bit complicated but the application remembers the windows settings between sessions.

The detection of APs is automatically done when a data file is opened. To avoid that this quite time-consuming filtering is done every time a data file is analyzed, the found APs are stored in a *project*. This project also stores information about threshold settings, MHT algorithm settings, found paths and their parameters. This makes it easy to continue a previously started analysis. Information about the data file that is connected to the project is displayed, see Figure 17.

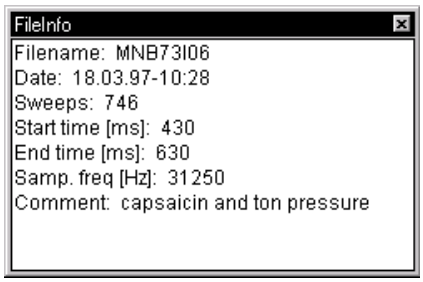


Figure 17: Information about the attached data file is displayed.

The data files that are analyzed may be more than hour long. If the whole data file is viewed, it is difficult to do any analysis. Therefore an infinitely variable zoom ability has been implemented. This makes it possible to study everything from a single trace to the whole file. The signals, action

* Docked means that the border of the window is removed and it is put into the main window.

potentials, found paths, and fitted curves are displayed in a scrollable view, see Figure 18. By default, the tracking is done on the data shown on the screen only.

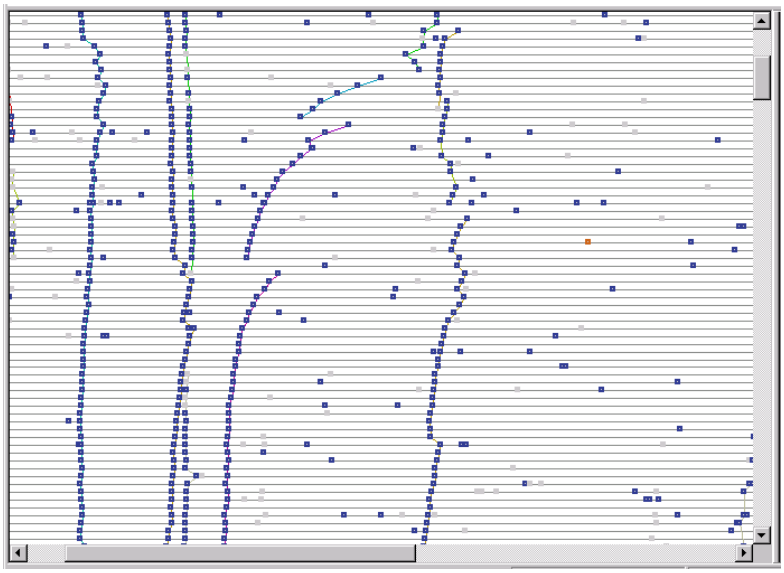


Figure 18: A scrollable view with infinite variable zoom abilities makes it possible to customize the viewed area. The view displays successive traces from top to bottom.

The user may choose which AP that shall be used in the tracking. This is done in a window that displays latency versus amplitude. A rough threshold is used in the matched filtering and it is possible to dynamically set a threshold above this. This window is separated from the main view area but they display the same set of AP (chosen in the main view).

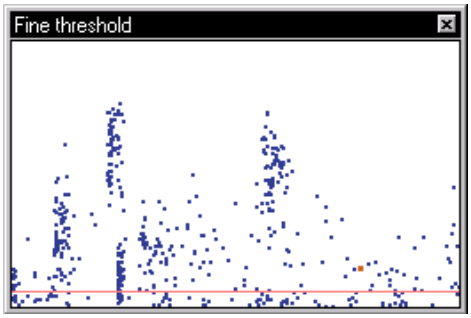


Figure 19: The threshold, seen as a red line, is set dynamically by the user.

When the filtering, tracking, and curve fitting is done, the resulting parameters are displayed in a window. The user may select a row in this window and the corresponding path will highlight in the other windows. Parameters from a row or the entire window are easily exported to another program for further processing. The parameters displayed are latency (y_0), latency shift (A), and recovery time constant (α), see Figure 20.

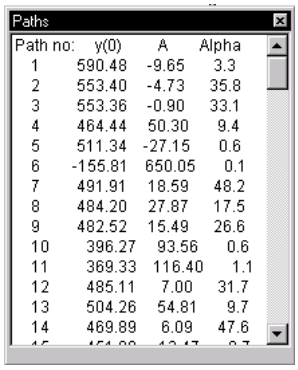


Figure 20: The parameters of the fitted curves are displayed in a table. The user may select a row and the corresponding path will highlight in the other views.

The analyst may change found paths manually. This is done exclusively with the mouse, see Figure 21. Found paths may also be completely deleted.

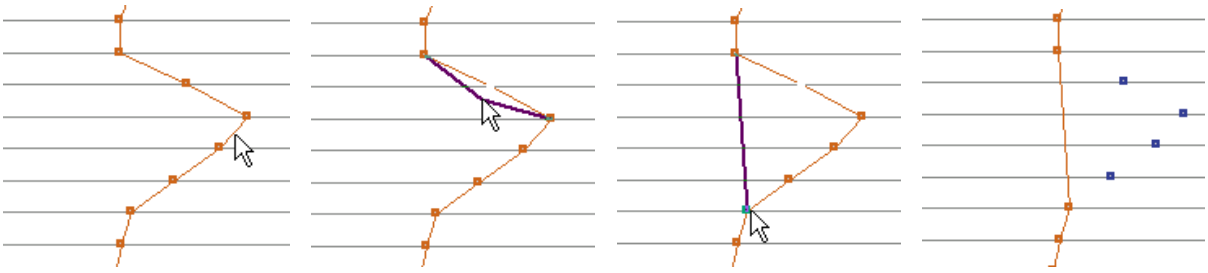


Figure 21: Editing a path that is considered wrong. To the left, the path has been selected. Then the user uses the mouse to change the path. The rightmost picture shows the resulting path.

Important information is displayed in the status bar on the bottom panel of the application. The information displayed is depending on what action the user takes. For example if a single AP is selected, the unique id, amplitude, and latency are displayed. Current threshold as well as the sweep and latency of the current mouse position are always displayed.

Figure 22: The status bar displays important parameters. The left box displays sweep and latency of the mouse pointer. The middle box shows the id, amplitude and latency of the selected AP. The right box shows current threshold settings.

The detection and the association algorithms are quite computationally heavy and may take a couple of minutes to complete. To inform the user about the current state and to avoid confusion, progress bars are used. The progress dialogs also allow the user to stop the calculations.

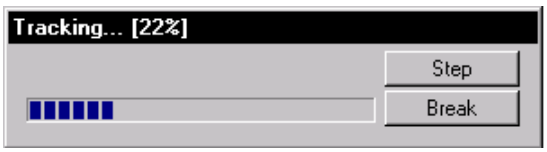


Figure 23: Progress bars are used during detection and tracking to inform the user about the current state.

The available commands are accessed by a standard menu. Some commonly used commands are also accessible via a toolbar, see Figure 24.

Figure 24: A toolbar gives access to the most important features.

A picture copy facility has been added to make it easy to publish pictures generated by the application. To allow customizations of the pictures, a number of different color schemes may be chosen. The default mode uses black background but it may be convenient to use, e.g., a white background when publishing. In the current application, the copied pictures will not include axes and scales but this feature will be added later if desired. It is also possible to print the view area directly from the application. A picture of the graphical user interface is included in appendix A1.

5 Example

This chapter uses a real recording to demonstrate the performance of the application. The MHT parameters that are used are shown in Figure 25. Some explanations are needed. Track combining is not used (nScan=0). The pruning logic is set to keep 64 hypotheses after processing each observation and 8 hypotheses after each trace cycle. The *initial probability of detection* is quite high, which means that tracks are punished if they miss initial update opportunities. The *max delta latency* variable is used before the Kalman filter is initiated. The Kalman filter needs two measurements to initiate which means that max delta latency is used prior to the addition of the second observation. The gate is fixed, with size 20. The *shortest track possible* is a parameter that determines the minimum length of the tracks that shall be used in the final result.

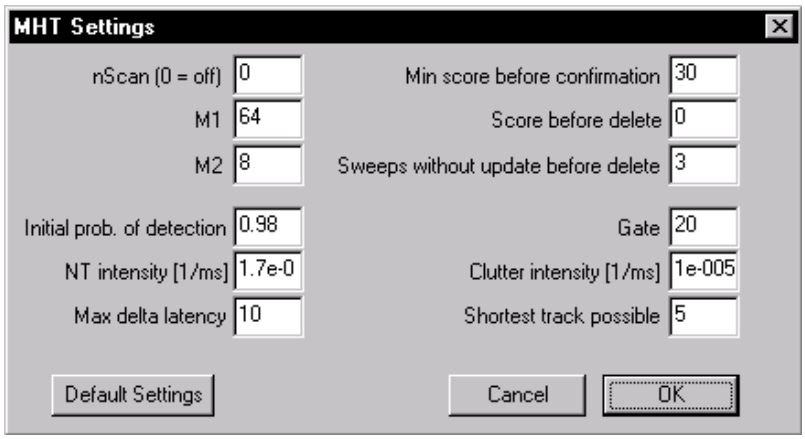


Figure 25: The MHT settings used in the example. Note the Default Settings button, which always makes it possible to reset to default configuration.

The area that is tracked contains four different fibers. One of the fibers reacts due to some additional stimulus and is activated. The activation is seen as a latency shift after the 530th trace. The fiber to the left doesn't react at all and remains at a steady latency. The two fibers to the right in figures 25-28 are probably sympathetic fibers, conducting signals to the skin. These fibers show an irregular behavior due to the brain's activation. The model that is used can't explain their latency course and the tracking algorithm isn't expected to be able to find these paths.

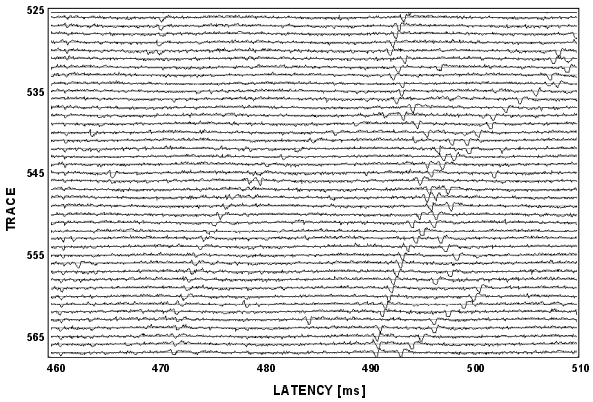


Figure 26: Successive raw signals recordings displayed from top to bottom.

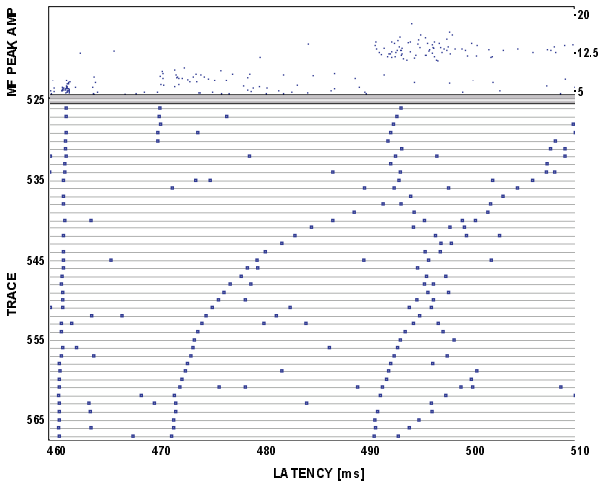


Figure 27: The detected APs. The top plot displays latency versus MF peak amplitude. The bottom plot displays latency versus trace numbers. The latency scale is equal in both plots.

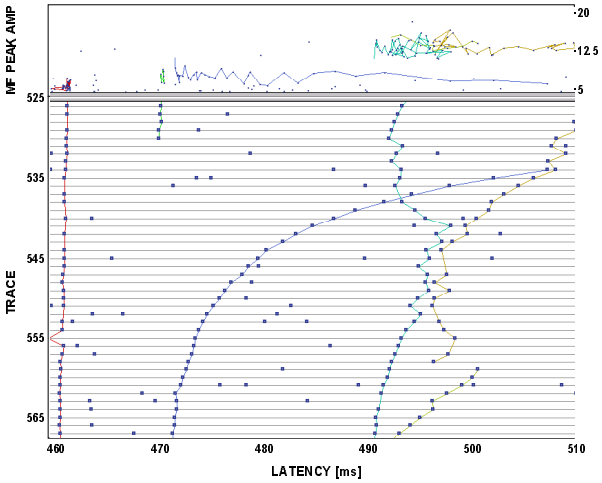


Figure 28: The paths found by the MHT algorithm. Note that the amplitude of the sympathetic fibers differ from the amplitude of the activated C-unit.

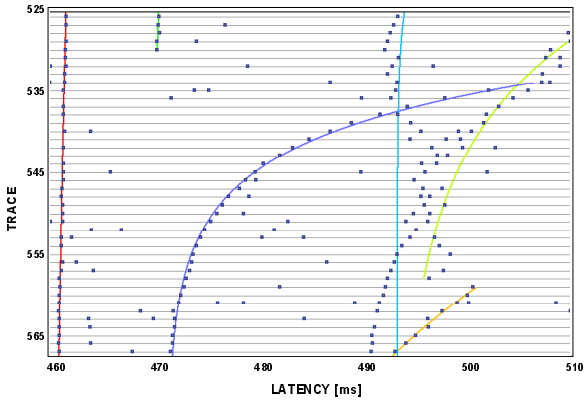


Figure 29: Curves are fitted to the different paths. Note that the green and dark blue paths belonging to the activated fiber has the same steady latency.

Note that the MHT algorithm tracks the path crossings correctly. In the top part of Figure 28 it is seen that the amplitude of the stimulated fiber is lower than the amplitude of both the sympathetic fibers. The tracking performance has probably been improved by utilizing this information in the filtering.

The parameters of the curves in Figure 29 are displayed in Table 5. The model is given by $y(t) = y_0 + Ae^{-\alpha_0(t-t_0)}$, see Figure 5. As mentioned before, the exponential model does not work for the sympathetic fibers and they could be omitted. Note that the estimated y_0 of the both paths belonging to the activated fiber, is similar.

y_0	A	α	
454.84	1.96	3.1	The fiber that doesn't react at all
465.05	2.40	8.3	The activated unit before activation
467.22	37.82	32.9	The recovery of the activated unit
467.94	31.18	9.6	The short, right bottom path of the sympathetic fiber
490.99	13.42	41.8	The left most sympathetic fiber
491.30	22.68	17.0	The sympathetic fiber to the right

Table 5: Parameters of the found tracks.

6 Discussion

Early experiments indicated that the chosen algorithms would be a good solution to the automatic analysis problem. Nevertheless, it has been a challenge to develop the automatic analysis tool. Many parts had to be united and the resulting application should be fairly easy to use. Some questions that we had at the beginning of the development were: Would the tracking work smoothly on an ordinary personal computer? Would it be efficient to use object oriented programming to solve the problems?

The achieved results correspond well with what an analyst would consider correct. The application simplifies the analysis and minimizes the required manual work. The tracking is quite fast and not very memory consuming. The application allocates approximately 4 MB of memory depending on the data file that is analyzed. During tracking, the memory requirements sometimes rise above 5 MB. The allocated memory is much dependent on the number of APs in each sweep.

By using object oriented programming the code is quite easy to understand and has the speed of common C-code. One drawback with the MHT algorithm is that there are some parameters to adjust, e.g., false and new target densities, gate, and initial probability of detection. A drawback with the Kalman prediction technique used is that the filtering performance is dependent on the recovery time constant α. Perhaps it will be necessary to replace the filter by an IMM based predictor, using multiple values of α.

By using a tree node structure to store tracks, it is possible to improve the efficiency of the MHT algorithm [9]. The idea is to implement a tree where a track and a list of pointers to its children form a node. The tracks in the most probable hypothesis are traced back for N observations and the ancestor track becomes a root node. All confirmed tracks that are not descendants of one of these root nodes will be deleted. This means that this logic uses current data to make irrevocable decisions regarding the correct hypotheses N traces back in time. This logic reduces the number of tracks required in typical scenarios by about half when compared to the algorithms described in chapter 3. This tree logic may be implemented if the association procedure is considered too slow.

Hopefully this application will help the researchers in their pain studies. The development of the application will probably continue as the analysts ask for extra features. Perhaps the application will be used in more ways than originally intended.

7 Acknowledgements

I would like to thank M. Sc. Björn Hansson for helping me with the development of the application. I also would like to thank Prof. Erik Torebjörk and Dr. Med. Sci. Roland Schmidt for valuable comments about the neurophysiological background. Comments and suggestions from colleagues at the Department of Clinical Neurophysiology have also been useful to improve the design of the application. I am also very grateful to Prof. Mikael Sternad who assisted with the proofreading of the report.

This work was supported by the Swedish Medical Research Council, project no. 5206.

8 References

1. Hansson B, Forster C, Torebjörk E, “Matched filtering and multiple hypothesis tracking applied to C-fiber action potentials recorded in human nerves”, Proc. of Signal and Data Processing of Small Targets, SPIE, vol. 3373, 1998
2. Tortora, Grabowski, “Principles of anatomy and physiology”, Harper Collins, pp. 346-347
3. Hallin R G, Torebjörk H E, “Afferent and efferent C units recorded from human skin nerves in situ”, Acta Soc. Med. Ups. 75, 1970, pp. 277-281
4. Torebjörk H E, Hallin R G, “C-fibre units recorded from human sensory nerve fascicles in situ”, Acta Soc. Med. Ups. 75, 1970, pp. 81-84
5. Torebjörk H E, Hallin R G, “Responses in human A and C fibres to repeated electrical intradermal stimulation”, J. Neurol. Neurosurg. Psychiatry 37, 1974, pp. 653-664
6. Blackman S S, “Multiple-Target Tracking with Radar Applications”, Artech House, Dedham, MA, 1986.
7. Blackman S S, Dempster R J, Tucker G K, Roszkowski S H, “Application of Multiple Hypotheses Tracking to Shipboard IRST Tracking”, SPIE Vol. 2759, 1996, pp. 441-452.
8. Lerro D, Bar-Shalom Y, “Interacting Multiple Model Tracking with Target Amplitude Feature”, IEEE Trans. on Aerospace and Electronic Systems Vol. 29, 1993, pp. 494-508.
9. Blackman S S, Dempster R J, Tucker G K, Roszkowski S H, “Recent Multiple Hypothesis Tracker Advances and Results”, Hughes Aircraft Company, 1995