Multipath Time Delay Estimation and Registration for a Moving Contact with Interference

A Thesis submitted to the College of Graduate Studies and Research in Partial Fulfillment of the Requirements for the degree of Master of Science in the Department of Electrical Engineering University of Saskatchewan Saskatoon

by

Liet T. Vu

September 1995

© Copyright Liet T. Vu, 1995. All rights reserved.
Copyright

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain may not be allowed without my written permission. It is also understood that due recognition will be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Electrical Engineering
University of Saskatchewan
Saskatoon, Saskatchewan S7N 0W0
Abstract

The objective of this work is to develop effective algorithms for Time Difference of Arrival Estimation and Time Delay path pair association for multiple moving sources in the presence of other interfering sources in a multipath environment using pairs of omnidirectional receivers. The work will focus on the problem of two different sources moving on straight line paths past two receivers.

Received signals are simulated based on the assumptions that the source signals are stationary, uncorrelated, zero mean, Gaussian random signals. For simplicity two signal paths are considered in this work; direct and surface bounce paths. Interpolation is used to create the time companding effect at the received signal. Ambient noise at the receiver’s filter is also simulated and added to the received signal.

An expression is developed for the expected value of the time cumulative average correlation of the two received signals for a single moving source in a multipath environment. This expression is further developed to enhance the multiple moving sources case. The resulting equation is a function of spectral densities of the two source signals. Reasonable simplifying assumptions are used to produce a more manageable expression for the expected value of the time average cross-correlation.

The Time Difference of Arrival (TDOA) estimation algorithm is developed using the deskewed short-time correlator technique. The desirable property of this technique is “correlate first, then compensate” which has the advantage of significant computational reduction compared to other techniques [1]. An alternative algorithm, which uses the Select-Correlate-Sum technique [2], is also proposed. The alternative algorithm is similar to the proposed technique where the main difference is in the sequence of execution. Simulated results indicate both algorithms provide similar
performance where the proposed algorithm has the advantage of computational time reduction and simplicity of implementation.

The TDOA estimates registration algorithm uses the relationships among the TDOA estimates obtained from the autocorrelation and cross-correlation to classify the TDOA estimates as belonging to one source or the other. Furthermore, the ascending order in time of the TDOA estimates are used to register the TDOA estimates to the pairs of paths that produce each TDOA estimate.
Acknowledgements

I would like to thank Dr. Daku for his supervision of this work. His assistance and advice, during the research and the preparation of the thesis, are greatly appreciated.

I wish to acknowledge the financial support of the Department of National Defence, especially Defence Research Establishment Atlantic, during the research and preparation for this thesis.

I wish to thank my sister, Nancy, my brother in law, David, and my wife, Hong Nga, whose encouragement, support and patience made the completion of this thesis possible.
# Table of Contents

Copyright i  
Abstract ii  
Acknowledgements iv  
Table of Contents v  
List of Figures viii  
List of Tables x  
List of Acronyms xi

1 Introduction 1  
  1.1 Background ................................................... 1  
  1.2 Problem Statement ............................................ 3  
  1.3 System Approach to the Problem ................................. 4

2 The Transmission System Model 7  
  2.1 Source Signal Model ........................................... 7  
  2.2 Source and Sensor Geometry ................................... 8  
  2.3 Model of the Ocean Channel .................................. 10  
  2.4 The Transmission System Model ................................. 13  
  2.5 Assumptions in the Received Signal Simulation ................. 13

3 Mathematical Background 15  
  3.1 Standard Approach for TDOA Estimation ......................... 15  
  3.2 TDOA Estimation for A Single Moving Source .................. 18  
    3.2.1 Model of Linear Varying Time Delay ......................... 18  
    3.2.2 A Single Moving Source in a Multipath Environment ........ 22  
  3.3 TDOA Estimation for a Moving Source with Interference .......... 24  
  3.4 Summary ...................................................... 31
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A single source with two sensors in a multipath environment</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Model of the source signal</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Two dimensional diagram of a single source two Sensor configuration</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Model of the ocean channel</td>
<td>12</td>
</tr>
<tr>
<td>2.4</td>
<td>Model of the hydrophone</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Model of the source and transmission system</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>TDOA estimation using generalized correlation</td>
<td>17</td>
</tr>
<tr>
<td>3.2</td>
<td>A single moving source in a single path environment</td>
<td>18</td>
</tr>
<tr>
<td>3.3</td>
<td>Dividing the integration time period into short time intervals $T_d$ and</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>taking short segment $T_s$ in every $T_d$ for correlation</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>A single moving source with interference in a multipath environment</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>Plot of autocorrelation peak</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>Data sequence for correlation</td>
<td>37</td>
</tr>
<tr>
<td>4.3</td>
<td>Effect of noise and interference on correlation peak</td>
<td>42</td>
</tr>
<tr>
<td>4.4</td>
<td>Effect of interference on contact correlation peak</td>
<td>43</td>
</tr>
<tr>
<td>4.5</td>
<td>Block diagram of the short time cross-correlator for the Algorithm I</td>
<td>44</td>
</tr>
<tr>
<td>4.6</td>
<td>Block diagram of Algorithm I</td>
<td>45</td>
</tr>
<tr>
<td>4.7</td>
<td>Plot of the passive ambiguity surface for a single moving source with</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>interference</td>
<td></td>
</tr>
<tr>
<td>4.8</td>
<td>Block diagram of the short time cross-correlator for Algorithm II</td>
<td>47</td>
</tr>
<tr>
<td>5.1</td>
<td>Top view of the search space for a two-sensor configuration</td>
<td>49</td>
</tr>
<tr>
<td>5.2</td>
<td>Overlaying direct paths on surface paths</td>
<td>52</td>
</tr>
<tr>
<td>5.3</td>
<td>Block diagram for the TDOA estimates registration</td>
<td>53</td>
</tr>
<tr>
<td>6.1</td>
<td>Block diagram of the implementation of the ocean transmission model in</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>software</td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>Improvement in the SNR of the correlation peak with increased number of</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>accumulated segment correlograms</td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>Correlation peaks obtained by using Algorithm I</td>
<td>62</td>
</tr>
</tbody>
</table>
6.4 Correlation peaks obtained by using Algorithm II ..................... 63
6.5 Block diagram for the first model used in the implementation of the registration algorithm ................................. 65
6.6 Block diagram for the second model used in the implementation of the registration algorithm .............................. 68
A.1 Tree diagram of the programs in implementing the received signal simulation ................................................. 89
A.2 Tree diagram of the programs in implementing the TDOA estimation using cross-correlation ................................. 110
A.3 Tree diagram of the programs in implementing the TDOA estimation using autocorrelation .................................. 110
A.4 Tree diagram of the programs in implementing the TDOA estimates registration .............................................. 118
List of Tables

6.1 Comparison between actual values and simulated results for the first trial .............................................. 79
6.2 Comparison between actual values and simulated results for the second trial ........................................... 81
List of Acronyms

1. DSTC - Deskewed Short Time Correlator
2. SCS - Select Correlate Sum
3. TDOA - Time Difference of Arrival
4. PAS - Passive Ambiguity Surface
1. Introduction

The documented work in this thesis is an extension of previous works done in the passive sonar problem of underwater acoustic source localization. This work focuses on the problem of the time difference of arrival (TDOA) estimation and association of two sources moving in straight line paths with constant velocity past two sonar sensors.

1.1 Background

A moving source emits sound which comes from its moving mechanical parts such as engine, propeller or from the motion of water flowing due to the source movement [3]. When a source is detected in a certain region of the ocean, passive sonar sensors or hydrophones are then dropped in the ocean near the detected location. These sensors, which are attached to a float mechanism, extend to the preset depth of at most a few hundred meters. The float mechanism also contains a transmitter which continuously sends the information collected back to the platform, for example a ship, for analysis. These sensors are underwater acoustic transducers that receive the acoustic signal from the source. Information extracted from these received signal are used to determine the location of the source.

The sound signal radiates from the source in a spherical pattern at a speed, which changes primarily due to the variation in temperature and pressure with depth, and secondarily to other factors such as salinity [3], but it is assumed to be constant in this region of the ocean. The sound signal propagates to the sensors over different paths, direct and reflected. An ocean environment which includes reflected paths is
called a multipath environment. The source is assumed to be in deep water since this is the main application for this type of localization. When the source is in deep water, the attenuation involving a bottom bounce will be relatively large thus only two paths are to be considered in this work, the direct and surface reflected paths. Figure 1.1 shows a two dimensional diagram of the path signals received by the sensors in a multipath environment. The notation for the path signals are $d_1$ for the direct path signal of sensor 1, $d_2$ for the direct path signal of sensor 2, $s_1$ for the surface reflected path signal of sensor 1 and $s_2$ for the surface reflected path signal of sensor 2.

The sound signal arrives at the sensors at different points in time over each path. For example, as shown in Figure 1.1, the sound signal will arrive at sensor 2 earlier over the direct path $d_2$ than at sensor 1 over the direct path $d_1$. The signal received over the path $d_1$ can be considered a delayed version of the signal received over the path $d_2$. The amount of this delay is known as the time difference of arrival (TDOA) and in this case particularly for the paths, $d_1$ and $d_2$, it is called the path pair TDOA. This TDOA is the key information to be determined for the underwater acoustic source localization.

The sound signal received at the sensors can be converted into electrical signals which are sampled, digitized and then processed using digital computing resources. The TDOA can be found by cross-correlating the sample sequences [4] from the sensors. The correlation will examine the similarity between the sample sequences and produce a peak (if the similarity exists) at a time shift which corresponds to the TDOA or the difference in time delay of the received signals at the two sensors. For the case of a single moving source as shown Figure 1.1, cross-correlation of the sample sequences from the two sensors will show four peaks corresponding to the four TDOA estimates of the path signals between the two sensors, $d_1$ and $d_2$, $d_1$ and $s_2$, $s_1$ and $s_2$, and $s_1$ and $d_2$. The location of the source underwater can be found by the intersection of the surfaces defined by the TDOA estimates [5].
Figure 1.1: A single source with two sensors in a multipath environment

1.2 Problem Statement

The problem of TDOA estimation and association is relatively straightforward for a single stationary or slowly moving source in an ideal environment where only the TDOA peaks produced by the source are present. A more realistic environment, with multiple moving sources in a multipath environment, significantly complicates the problem.

A standard approach to the TDOA estimation is generalized correlation of the received signals [6]. Generalized correlation is not useful in the case where the source motion causes the TDOA to change significantly over one integration period. This causes smearing of the TDOA peaks in the correlation function. The top of the peaks will be spread out, making it more difficult to point out the maximum and also making the maximum more susceptible to dislocation by noise. The changing rate of the TDOA is referred to as the relative time companding [7], [8]. This relative time companding has to be compensated in order for the correlation to produce meaningful results.

The presence of the interference adds several more peaks in the correlation function of the received signal at the two sensors. The signal from two sources will affect each other in the correlation, the correlation peaks will be shifted and their position
on the axis of the correlation variable in the correlograms will not correctly represent
the corresponding TDOA estimates.

The TDOA path pair association problem becomes especially critical in mul-
titarget scenarios when the number of TDOA estimates can be very large and the
computational resources required to perform association are correspondingly large.
The object of this work is to develop fast, robust algorithms to perform this TDOA
estimation and association problem.

1.3 System Approach to the Problem

The problem of data association in broadband, multipath target tracking can
be broken into two subproblems: (1) estimating the TDOA, and (2) associating the
TDOA estimates to the pairs of paths that produced each TDOA.

The subject of time delay estimation has received considerable attention in the
past [9]; however, the literature dealing with multiple moving sources in a multipath
environment is not large. A variation of the generalized correlation technique, [10],
suggests compensating the relative time companding effect by time scaling the wave
form in one channel relative to the other, filtering the output accordingly and correlat-
ing the two signals to produce the same output as when no relative time companding
exists. The problem with this technique is time scaling is computationally expensive
and implementation is very complex.

Another generalized correlation-based technique, the Select-Correlate-Sum tech-
nique [2] developed by K. Jeffrey, suggests a different approach. Short time intervals
of signal data are extracted from the received signal data sequence, which are suffi-
ciently small that the time delay can be assumed constant. The compensation for the
relative time companding due to source motion can be accomplished by time shift-
ing the short time intervals. These short time intervals will be cross-correlated and
the resulting correlograms are accumulated to give the time average cross-correlation
followed by the TDOA estimation. An algorithm based on this technique is also
developed and suggested as an alternative to the proposed algorithm because this alternative algorithm is similar to the proposed algorithm.

A coherent technique which sums short-time correlograms, appropriately deskewed in time delay, has been proposed [11], [12]. Its computational advantage compared to the other techniques have been noted and simulated results indicate similar performance to that of the other techniques. The last approach, the Deskewed Short-Time Correlator (DSTC) [1], provides similar performance with significant reduction in computational time. The DSTC has the desirable property of "correlate first, then compensate", [11], [12] and [13]; cross-correlation is performed once and the relative time companding compensation is applied in sequential trials to the already cross-correlated data. The proposed TDOA estimation algorithm in this work is developed based on the DSTC technique.

The correlation peaks produced in the TDOA estimation process over a defined range of TDOA values and relative time companding can be represented by a surface called the Passive Ambiguity Surface (PAS). When two sources have the same TDOA estimate, the two peaks may merge together for a short period of time. Over a long period of time, the difference in velocity of the two sources will separate the TDOA peaks. If two sources have the same initial location and are moving at the same velocity, all the TDOA peaks of the two sources will be merged together in pair. Both sources may be detected as a single source.

An additional technique, the Interval-in-Interval correlation technique is introduced to be used in addition to other techniques in the TDOA estimation process. A smaller interval $T_s$ is taken within the short-time interval $T_d$ for correlation. This is done to improve the sharpness of TDOA peak relative to other peaks on the PAS, i.e., reducing the effect of interference, which makes it easier to detect the correct TDOA peaks.

A simple solution for associating the path pair TDOA estimates is by using the ascending order in time of the TDOA estimates. This can be proved using simple geometric theorems. An essential step in the TDOA estimates registration process
is to correctly classify the TDOA estimates to the corresponding source. Only by observing the positions and the rate of change of these peaks on the PAS, one might classify the peaks as belonging to one source or the other. This classification of the TDOA estimates can be done by using the relationships between the TDOA estimates obtained from the cross-correlation and autocorrelation of the received signals. After the TDOA estimates are classified to the corresponding source, the ascending order in time of the TDOA estimates will allow them to be registered to the correct pair of path signals that produced each TDOA estimate.

The simulation of the received signal is necessary for the verification of developed algorithms. The acoustic signal propagates from the source to the receivers through the ocean which is a very complex transmission medium that needs to be modelled [14]. The major items in the transmission medium considered here are the sound speed profile underwater, reflections, transmission loss and ambient noise. Reasonable assumptions are used to simplify the simulation process.

There are limitation in the developed algorithms for the TDOA estimation and registration process. The TDOA estimation algorithm developed is limited by environmental and geometric factors. The environmental factor includes the depth of the source where there is a change in sound speed due to variation in temperature and pressure with depth [3]. The geometric factor includes range and velocity of the source. These limitation factors are taken into account where reasonable assumptions are made in an attempt to simplify the problem.
2. The Transmission System Model

2.1 Source Signal Model

An underwater acoustic source generates noise that can be grouped into three classes: machinery noise, propeller noise, and hydrodynamic noise [3]. Machinery noise, which is self-explanatory, is caused by machinery within the source. Propeller noise is a hybrid having features common to both machinery and hydrodynamic noise. Hydrodynamic noise is caused by the irregular flow of water past the source. Machinery noise is due to mechanical vibration of moving parts within the source. The spectrum of machinery noise consists of a low level continuous spectrum containing strong tonal components. Propeller noise originates from outside the source in the form of cavitation noise induced by the rotating propellers. This propeller noise increases dramatically above a certain speed. The critical speed for a typical source is 3 to 5 knots. Propeller noise produces a continuous spectrum with a peak occurring between 100 and 1000 Hz; with increased speed the peak shifts to lower frequencies. The spectrum also contains some tonal lines. Hydrodynamic noise is usually small and marked by machinery and propeller noise. It is basically caused by pressure fluctuations associated with irregular flow of water past the source.

The underwater acoustic source, which produces the above types of noise, is modelled as a point signal source with an uniform omnidirectional radiation pattern. The assumption of a point noise source is reasonable for radial distances much larger than the size of the source. It is assumed that the signal produced by the source has a continuous broadband spectrum. This is a reasonable assumption for a source travelling at a speed higher than 5 knots. A block diagram of the model used to represent the physical source is shown in Figure 2.1. A white noise generator produces a
signal, $w(t)$, which is filtered by a filter $H_s(f)$ to produce the signal emanating from the source, $s_p(t)$, where the $p$ subscript is associated with the pressure wave.

### 2.2 Source and Sensor Geometry

The time difference of arrival (TDOA) information is determined by considering the sensors pairwise, thus, it is possible to view the multiple sensor scenario as multiple sets of two sensors. Figure 2.2 shows a simplified diagram of a single-source two-sensor configuration in a multipath environment. A Cartesian coordinate system is used, with the origin on the ocean surface. The $xy$ plane is the ocean surface and the positive $z$ direction is down. This figure shows the direct and surface bounce paths for a single moving source. The two sensors referred to as $d1$ and $d2$ are located directly under the $x$ axis at a depth of $z_r$ below the surface with a spacing of $2x_r$ between them. The locations of sensor $d1$ and sensor $d2$, expressed as vectors in the Euclidean space respectively, are

$$x_{d1} = [x_{d1}, y_{d1}, z_{d1}]^T = [-x_r, y_r, z_r]^T$$

$$x_{d2} = [x_{d2}, y_{d2}, z_{d2}]^T = [x_r, y_r, z_r]^T$$

where $y_r$ can be set to zero without loss of generality. The coordinate origin is halfway between the sensors on the ocean surface. With the symmetry, the positive $x$ and positive $y$ directions are arbitrary. Each of the sensors receives signals from a single source, or contact, located at $x_c = [x_c, y_c, z_c]^T$ by way of a direct path and a surface reflected path. The surface reflection is associated with a virtual sensor which
Figure 2.2: Two dimensional diagram of a single source two Sensor configuration
is located directly above the actual sensor at a height of $-z_r$. The locations of the virtual sensors $s1$ and $s2$ respectively are

$$x_{s1} = [x_{s1}, y_{s1}, z_{s1}]^T = [-x_r, 0, -z_r]^T \tag{2.3}$$

$$x_{s2} = [x_{s2}, y_{s2}, z_{s2}]^T = [x_r, 0, -z_r]^T \tag{2.4}$$

The distance from the source to sensor $i$, as shown in Figure 2.2, is denoted by

$$l_i = ||x_c - x_i|| = \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2 + (z_c - z_i)^2} \tag{2.5}$$

where $i$ can take on values $d1$, $d2$, $s1$ and $s2$.

### 2.3 Model of the Ocean Channel

The acoustic signal propagates from the source to the receiver through the ocean which is a very complex acoustical transmission medium. The major items considered here in the transmission medium are the sound speed profile, reflections, transmission loss and ambient noise level. This complexity can be greatly reduced with assumptions.

Generally there is a change in sound speed with depth [3], which is due primarily to variation in temperature and pressure with depth, and secondarily to other factors such as salinity. This sound speed variation causes the various source-receiver paths to curve which changes the path lengths and complicates the procedure of locating the source. It is assumed that the source is moving on a straight line with little change in depth. The source is assumed to be in deep water, since this is the main application area for this type of localization. Furthermore, the source is assumed to be in the near vicinity of the sensors. Thus, the assumption of constant sound speed profile is reasonable since over short distances the propagation path is nearly a straight line.

When the source is in deep water, the attenuation of any paths involving a bottom bounce will be relatively large [14], thus only two paths of the source signal
are to be considered in this work, the direct and surface reflected paths. For simplicity all the losses in transmitting the signal are combined into a frequency independent attenuation constant which includes spherical spreading loss, ocean absorption loss and, for the surface bounce path, the loss resulting from a surface reflection and the phase reversal at the surface boundary.

The above information and the source-sensor configuration are used to generate the ocean channel model, which is shown in the block diagram in Figure 2.3. The source generates a signal, \( s_p(t) \). Due to the source motion, the signal received at each sensor will be Doppler scaled (time compressed or expanded) leading to a relative time companding \([7],[8] \). The source approaches the sensor with a different velocity over each path; thus, the relative time companding over each path is different. This relative time companding is derived in Chapter 3 using a Taylor's series expansion and is denoted \( E_d \) and \( E_s \) for direct and surface path, respectively.

The signal received at each sensor is appropriately delayed for the surface path and the direct path (assuming constant sound speed profile) by

\[
D_s = \frac{\text{length of surface path}}{\text{speed of sound}}
\]

\[
D_d = \frac{\text{length of direct path}}{\text{speed of sound}}
\]

The delayed signals are multiplied by the transmission loss attenuation coefficients \( g_s \) and \( g_d \). The resulting surface and direct signals are added to the ambient noise \( n_p(t) \) to give the signal received at the sensor, \( r_p(t) \). The ambient noise, as previously described, can be modelled as a random point source followed by a shaping filter, resulting in a continuous arbitrary broadband spectrum for the noise.

The hydrophone is a transducer that converts acoustical energy to electrical energy \([14] \). A model of the hydrophone is shown in the block diagram of Figure 2.4. The pressure signal \( r_p(t) \) is converted to an electrical signal \( r(t) \) by multiplying with a constant \( K \). \( K \) has units volts/\( \mu Pa \), \( r_p(t) \) has units \( \mu Pa \), and \( r(t) \) has units volts.
Figure 2.3: Model of the ocean channel
2.4 The Transmission System Model

A complete model for the physical transmission system from the source to the receiver, which is a summary of the previous sections, is shown in Figure 2.5. This figure includes models of both of the receivers.

2.5 Assumptions in the Received Signal Simulation

This is a summary of the assumptions made in simulating the received signal:

- Constant sound speed underwater
- The source is moving on a straight line past two receivers
- Constant source velocity
- Source signal is modelled as a point noise source
- Source signal is white noise, band limited
- All losses, such as transmission loss, spreading loss, surface reflected loss etc., are combined into a single attenuation constant
- Ocean surface is flat with perfect reflection of the surface reflected path
- Only direct and surface reflected paths are considered

These assumptions are made in an attempt to simplify the problem of simulating the received signal. The received signal simulation is implemented using MATLAB programming [15]. The simulation will be described in more details in Section 6.1 and the software code is included in Appendix A.1.
Figure 2.5: Model of the source and transmission system
3. Mathematical Background

This chapter establishes the mathematical background for the TDOA estimation. Mathematical expressions for the TDOA estimates are derived from the simple case, a single stationary source in a multipath environment, to the case of multiple moving sources in a multipath environment. These expression provide some insights into the development of the correlation technique that is used in the TDOA estimation.

3.1 Standard Approach for TDOA Estimation

Consider the case of a single stationary source in a multipath environment as shown in Figure 2.2. The received signals at the sensors are given by

\[ r_1(t) = g_{d_1} s(t - D_{d_1}) + g_{s_1} s(t - D_{s_1}) + n_1(t), \text{ and} \]
\[ r_2(t) = g_{d_2} s(t - D_{d_2}) + g_{s_2} s(t - D_{s_2}) + n_2(t) \]

where

- \( r_m(t) \) is the signal received by the \( m^{th} \) sensor,
- \( s(t) \) is a random, broadband signal produced by the source,
- \( n_m(t) \) is the filtered noise received by the \( m^{th} \) sensor, \( s(t) \) and \( n_m(t) \) are assumed to be stationary, zero mean, uncorrelated, Gaussian random processes [16],
- \( g_{d_m} \) and \( g_{s_m} \) are the attenuation coefficients, which are constant for a stationary source, for the direct and surface path, respectively, from the source to the \( m^{th} \) sensor. Note that \( g_{s_m} \) includes the reflection loss for the surface path.
The desired information in the received signals is the Time Difference of Arrival (TDOA) of various paths between the two sensors. For example if the direct paths, \( l_{d1} \) and \( l_{d2} \), are selected, the TDOA is \( D_{d1d2} = D_{d1} - D_{d2} \). A standard approach to estimate the TDOAs between the various paths, the Generalized Correlation Method, [6], [10]. is to take the time average correlation of the received signals after appropriate filtering. The cross-correlation of the two received signals \( r_1(t) \) and \( r_2(t) \) is given by

\[
R_{12}(\tau) = E[r_1(t)r_2(t-\tau)]
\]

\[
= E[g_{d1}g_{d2}s(t-D_{d1})s(t-\tau-D_{d2}) + g_{s1}g_{s2}s(t-D_{s1})s(t-\tau-D_{s2}) + g_{s1}g_{s2}s(t-D_{s1})s(t-\tau-D_{s2}) + g_{d1}g_{d2}R_{ss}(\tau-D_{d1d2})]
\]

\[
= 
\begin{align*}
&= g_{d1}g_{d2}R_{ss}(\tau-D_{d1d2}) + g_{d1}g_{s2}R_{ss}(\tau-D_{d1s2}) + \\
&= g_{s1}g_{d2}R_{ss}(\tau-D_{s1d2}) + g_{s1}g_{s2}R_{ss}(\tau-D_{s1s2})
\end{align*}
\]

(3.3)

where \( R_{ss}(\tau) \) is the autocorrelation of the source signal, \( s(t) \), \( E[.\] represents the expected value and \( D_{jk} \) is the differential time delay between the path-pair signals, \( j = d1,s1 \) and \( k = d2,s2 \), for example \( D_{d1d2} = D_{d1} - D_{d2} \). The cross-correlation function is estimated with the time average

\[
\hat{R}_{12}(\tau) = \frac{1}{T} \int_{0}^{T} r_1(t)r_2(t-\tau)dt,
\]

\[
\begin{align*}
&= \frac{1}{T} \int_{0}^{T} [g_{d1}g_{d2}s(t-D_{d1})s(t-\tau-D_{d2}) + g_{d1}g_{s2}s(t-D_{d1})s(t-\tau-D_{s2}) + g_{s1}g_{s2}s(t-D_{s1})s(t-\tau-D_{s2}) + g_{d1}g_{d2}R_{ss}(\tau-D_{d1d2})]
\end{align*}
\]

(3.4)

where \( T \) is assumed sufficiently large to satisfy \( TB >> 1 \), and \( B \) is the bandwidth of the signal, \( s(t) \). The expected value of the time average cross-correlation is

\[
E[\hat{R}_{12}(\tau)] = \frac{1}{T} \int_{0}^{T} E[g_{d1}g_{d2}s(t-D_{d1})s(t-\tau-D_{d2}) + g_{d1}g_{s2}s(t-D_{d1})s(t-\tau-D_{s2}) + g_{s1}g_{s2}s(t-D_{s1})s(t-\tau-D_{s2})]dt
\]

\[
= \frac{1}{T} \int_{0}^{T} [g_{d1}g_{d2}R_{ss}(\tau-D_{d1d2}) + g_{d1}g_{s2}R_{ss}(\tau-D_{d1s2}) + g_{s1}g_{d2}R_{ss}(\tau-D_{s1d2}) + g_{s1}g_{s2}R_{ss}(\tau-D_{s1s2})]dt
\]

\[
= \frac{1}{T} [g_{d1}g_{d2}R_{ss}(\tau-D_{d1d2}) + g_{d1}g_{s2}R_{ss}(\tau-D_{d1s2}) + g_{s1}g_{d2}R_{ss}(\tau-D_{s1d2}) + g_{s1}g_{s2}R_{ss}(\tau-D_{s1s2})]
\]
The expression of the expected value of the time average cross-correlation indicates that $R_{12}(\tau)$ should have four distinct peaks, at shifts of $D_{d1d2}$, $D_{d1s2}$, $D_{s1d2}$, and $D_{s1s2}$. An sample result of implementing Equation 3.5 using MATLAB [15] is shown in Figure 3.1. This figure illustrates two negative and two positive peaks. The positive peaks are at $D_{d1d2}$ and $D_{s1s2}$ and the two negative peaks are at $D_{s1d2}$ and $D_{d1s2}$. The reason for the two negative peaks is due to the phase reversal of the two surface bounce path signals that are cross-correlated with direct path signals.
3.2 TDOA Estimation for A Single Moving Source

3.2.1 Model of Linear Varying Time Delay

The generalized correlation method is not very useful to estimate the TDOA for a moving source where source motion causes the TDOA to change significantly over one integration period, $T$. The source will cause smearing of the TDOA peaks in the correlation function. In this analysis only the direct paths are considered to simplify mathematical expressions. A diagram of a single moving source in a single path environment is shown in Figure 3.2. A general model of the received signal at the $m^{th}$ sensor is given by

$$r_m(t) = g_{dm}(t)s(t - D_{dm}(t)) + n_m(t), \quad (3.6)$$

The source movement is reflected in the time-varying signal transit time delays. Each sensor receives the signal, the direct path waveform with time-varying attenuation coefficient, $g_{dm}(t)$, and time varying delay, $D_{dm}(t)$. The source signal, $s(t)$, and ambient noise, $n_m(t)$, are assumed to be stationary, uncorrelated, zero mean, Gaussian random processes. The variation in the attenuation coefficients will be small over the
distance travelled by a source; therefore, they are assumed to be constant and take
on their value at \( t = 0 \), i.e., \( g_{dm} = g_{dm}(0) \). The time varying delay, \( D_{dm}(t) \), can be
expressed as

\[
D_{dm}(t) = \frac{l_{dm}(t)}{c}, \quad m = 1, 2. \quad (3.7)
\]

where \( l_{dm}(t) \) is the time varying distance of the path from the source to the \( m \)th sensor
and \( c \) is the speed of sound in water. Using a Taylor series expansion about \( t = 0 \)
and neglecting all higher order terms, the varying time delay is given by

\[
D_{dm}(t) = \frac{l_{dm}(0)}{c} + \frac{1}{c} \frac{dl_{dm}(t)}{dt} \bigg|_{t=0} t
\]

\[= D_{dm} + \frac{V_{dm}}{c} t \quad (3.8)\]

where \( D_{dm} = \frac{l_{dm}(0)}{c} \) is the delay at time \( t = 0 \) and \( V_{dm} \) is the velocity component of
the source in the direction of the \( m \)th sensor at time \( t = 0 \). The received signal at the
sensors can be written as

\[
r_m(t) = g_{dm} s(t - \frac{V_{dm}}{c} t - D_{dm}) + n_m(t), \quad (3.9)
\]

Since \( n_m(t) \) and \( s(t) \) are uncorrelated, the time average cross-correlation of the re-
ceived signals at sensor 1 and sensor 2 over the time interval \([0, T]\) is given by

\[
\hat{R}_{12}(\tau) = \frac{1}{T} \int_0^T r_1(t) r_2(t - \tau) dt
\]

\[= \frac{1}{T} \int_0^T g_{d1} g_{d2} s(t - \frac{V_{d1}}{c} t - D_{d1})
\]

\[s(t - \tau - \frac{V_{d2}}{c} (t - \tau) - D_{d2}) dt \quad (3.10)\]

The expected value of the time average cross-correlation is

\[
E[\hat{R}_{12}(\tau)] = \frac{1}{T} \int_0^T g_{d1} g_{d2} R_{ss}[\tau - \frac{V_{d2}}{c} \tau + t \frac{V_{d2} - V_{d1}}{c} + D_{d2} - D_{d1}] dt \quad (3.11)
\]

The cross-correlation \( R_{ss}[\cdot] \) becomes a maximum only if its argument is zero.
That is

\[
\tau - \frac{V_{d2}}{c} \tau + t \frac{V_{d2} - V_{d1}}{c} + D_{d2} - D_{d1} = 0 \quad (3.12)
\]
At time $t=0$

$$\tau_o - \frac{V_{d2}}{c} \tau_o + D_{d2} - D_{d1} = 0$$

and thus

$$\tau_o = \frac{D_{d1} - D_{d2}}{1 - \frac{V_{d2}}{c}}$$

(3.13)

where $\tau_o$ is the initial differential time delay at time $t=0$. At time $t > 0$, there will be an additional term, the relative time companding (differential velocity) $\frac{V_{d2}-V_{d1}}{c}t$, [8], [17], in the argument of $R_{ss}[\cdot]$. For the argument to be zero the time axis has to be compensated for the relative time companding $\frac{V_{d2}-V_{d1}}{c}$ by a factor $\beta(t)$ so that $\beta(t) - \frac{V_{d2}-V_{d1}}{c}t = 0$ which gives

$$\beta(t) = \frac{V_{d2} - V_{d1}}{c}t$$

The received signal has to be time-scaled for the cross-correlation to produce meaningful results. This problem can be overcome by using the short time interval correlation technique which proposes breaking up the integration interval, $[0, T]$, into a number of smaller intervals $[0, T_d]$, over which the time delay can be assumed constant. These smaller intervals are taken every $T_d$ seconds, such that $NT_d = T$ and $N$ is a preset number. This is illustrated in Figure 3.3. These smaller intervals are cross-correlated and then summed to give the time average cross-correlation.

The technique of taking a small interval $T_d$ within short time interval $T_d$ will be referred to as the interval-in-interval technique. This technique is used to improve the sharpness of resulting correlation peaks which will be explained later. The compensation term, $\beta(\gamma_d)$ where $\gamma_n = -T/2 + \frac{2n-1}{2}T_d$, $n = 1, 2, ..., N$ and $N$ is odd, is approximately constant over the short time interval $T_d$. Thus linear varying time delay can be assumed and the relative time companding $\beta(\gamma_d)$ can be set as

$$\beta(\gamma_d) = \gamma_d \epsilon,$$

(3.14)

where $\epsilon$ is a constant. With this approximation, and neglecting the noise terms, the signal model becomes:

$$r_1(t) = g_{d1}s(t - \frac{V_{d1}}{c}t - D_{d1}),$$

(3.15)
Figure 3.3: Dividing the integration time period into short time intervals $T_d$ and taking short segment $T_s$ in every $T_d$ for correlation

$$r_{2sc}(t) = g_{d2}s(t - \frac{V_{d2}}{c} t - D_{d2} - \gamma_d \epsilon)$$

(3.16)

where $r_{2sc}(t)$ is a compensated version of $r_2(t)$. The received signals at both sensors are subjected to relative time companding caused by the source motion, but it is the differential time delay that can only be estimated. Therefore, only one signal has to be compensated for the relative time companding in the TDOA estimation. It is $r_2(t)$ for this case. The time average cross-correlation of short time intervals of $r_1(t)$ and $r_{2sc}(t)$ is given by

$$\hat{R}_{12sc}(\tau) = \sum_{n=1}^{N} \frac{1}{T_s} \int_{\gamma_d - \frac{T_s}{2}}^{\gamma_d + \frac{T_s}{2}} r_1(t) r_{2sc}(t - \tau) dt,$$

$$\hat{R}_{12sc}(\tau) = \sum_{n=1}^{N} \frac{1}{T_s} \int_{\gamma_d - \frac{T_s}{2}}^{\gamma_d + \frac{T_s}{2}} (g_{d1}s(t - \frac{V_{d1}}{c} t - D_{d1})$$

$$g_{d2}s(t - \tau - \frac{V_{d2}}{c} (t - \tau) - D_{d2} - \gamma_d \epsilon)) dt$$

$$= \sum_{n=1}^{N} \frac{1}{T_s} \int_{\gamma_d - \frac{T_s}{2}}^{\gamma_d + \frac{T_s}{2}} g_{d1}g_{d2}[s(t - \frac{V_{d1}}{c} t - D_{d1})$$

$$s(t - \tau - \frac{V_{d2}}{c} (t - \tau) - D_{d2} - \gamma_d \epsilon)) dt$$

(3.17)

where $T_s \leq T_d$. The expected value of this time average cross correlation function is:

$$E[\hat{R}_{12sc}(\tau)] = \sum_{n=1}^{N} \frac{1}{T_s} \int_{\gamma_d - \frac{T_s}{2}}^{\gamma_d + \frac{T_s}{2}} g_{d1}g_{d2} R_{ss}[\tau(1 - \frac{V_{d2}}{c}) +$$
\[ \frac{V_{d_2} - V_{d_1}}{c} t + D_{d_2} - D_{d_1} + \gamma_d \epsilon \] 

In the expression for \( E[\hat{R}_{12_{sc}}(\tau)] \), if there are values of \( \tau \) and \( \epsilon \) that make the argument of \( R_s(.) \) zero for all \( t \) in the interval of integration, then these values of \( \tau \) and \( \epsilon \) maximize the expected value. That is

\[ \tau (1 - \frac{V_{d_2}}{c}) - \frac{V_{s_1} - V_{d_2}}{c} t + D_{d_2} - D_{d_1} + \gamma_d \epsilon = 0 \]

which gives

\[ -(\frac{V_{d_1} - V_{d_2}}{c}) t + \gamma_d \epsilon = 0, \text{ i.e.} \]

\[ \epsilon = \frac{V_{d_2} - V_{d_1}}{c} \]  

(3.19)

and

\[ \tau (1 - \frac{V_{d_2}}{c}) + D_{d_2} - D_{d_1} = 0, \text{ i.e.} \]

\[ \tau = \frac{D_{d_1} - D_{d_2}}{1 - \frac{V_{d_2}}{c}} \]  

(3.20)

With the relative time companding \( D_{d1d2} = D_{d_1} - D_{d_2} \) and \( V_{d_2} << c \), \( \tau \) can be approximated as \( \tau \approx D_{d1d2} \). This value of \( \tau \approx D_{d1d2} \), with the corresponding value of \( \epsilon \) compensating for the relative time companding \( \frac{V_{d_1} - V_{d_2}}{c} \), is the TDOA estimate for the path pair \( d1d2 \) of the received signals at the two sensors.

### 3.2.2 A Single Moving Source in a Multipath Environment

For a single moving source in a multipath environment, the case for single path can be extended by adding the surface bounce paths as shown in Figure 2.2. Using the same approximation as in the case for a single moving source in a single path environment, the received signals at sensor 1 and sensor 2 are given by

\[ r_1(t) = g_{d_1}s(t - E_{d_1} - D_{d_1}) + g_{s_1}s(t - E_{s_1} - D_{s_1}) + n_1(t), \text{ and} \]

(3.21)

\[ r_2(t) = g_{d_2}s(t - E_{d_2} - D_{d_2}) + g_{s_2}s(t - E_{s_2} - D_{s_2}) + n_2(t), \]

(3.22)

where
$g_{dm}$ and $g_{sm}$ are the attenuation coefficient for the direct and surface path of the source to the $m^{th}$ sensor respectively where $m = 1, 2$ and the subscript $d$ denotes the direct path and $s$ denotes the surface bounce path,

$E_{dm} = \frac{V_{dm}}{c}$ and $E_{sm} = \frac{V_{sm}}{c}$ are the time companding for the direct and surface path of the contact source, respectively,

$D_{dm}$ and $D_{sm}$ are the time delay for the direct and surface path of the source to the $m^{th}$ sensor, respectively,

The expected value of the time average cross-correlation is:

$$E[R_{12}(\tau, \epsilon)] = E\left[\sum_{n=1}^{N} \frac{1}{T_s} \int_{\nu_{d-d} - \frac{T_d}{2}}^{\nu_{d+d} + \frac{T_d}{2}} r_1(t)r_2(t - \tau - \gamma_d \epsilon) dt\right]$$

$$= E\left[\sum_{n=1}^{N} \frac{1}{T_s} \int_{\nu_{d-d} - \frac{T_d}{2}}^{\nu_{d+d} + \frac{T_d}{2}} \{g_{d1}s(t - E_{d1}t - D_{d1}) +
\quad g_{s1}s(t - E_{s1}t - D_{s1}) + n_1(t)\} \{g_{d2}s(t - \tau - E_{d2}t + \gamma_d \epsilon - D_{d2})
\quad + g_{s2}s(t - \tau - E_{s2}t + \gamma_d \epsilon - D_{s2}) + n_2(t)\} dt\right]$$

$$= E\left[\sum_{n=1}^{N} \frac{1}{T_s} \int_{nT_d + \frac{T_d}{2}}^{(n+1)T_d + \frac{T_d}{2}} \{g_{d1}g_{d2}R_{ss}(-E_{d1}t + \gamma_d \epsilon + \tau - D_{d1}t)
\quad + g_{d1}g_{s2}R_{ss}(-E_{d1}t + \gamma_d \epsilon + \tau - D_{d1}s)
\quad + g_{s1}g_{d2}R_{ss}(-E_{s1}t + \gamma_d \epsilon + \tau - D_{s1}t)
\quad + g_{s1}g_{s2}R_{ss}(-E_{s1}t + \gamma_d \epsilon + \tau - D_{s1}s)\} dt\right]$$

$$= E\left[\sum_{n=1}^{N} \frac{1}{T_s} \int_{nT_d + \frac{T_d}{2}}^{(n+1)T_d + \frac{T_d}{2}} \{g_{d1}g_{d2}R_{ss}(-E_{d1}t + \gamma_d \epsilon + \tau - D_{d1}t)
\quad + g_{d1}g_{s2}R_{ss}(-E_{d1}t + \gamma_d \epsilon + \tau - D_{d1}s)
\quad + g_{s1}g_{d2}R_{ss}(-E_{s1}t + \gamma_d \epsilon + \tau - D_{s1}t)
\quad + g_{s1}g_{s2}R_{ss}(-E_{s1}t + \gamma_d \epsilon + \tau - D_{s1}s)\} dt\right],$$

where
$E_{jk}$ is the differential time delay rate for the various paths of the source, $j = d1, s1$ and $k = d2, s2$, for example $E_{d1d2} = E_{d1} - E_{d2}$.

$D_{jk}$ is the differential time delay for the various paths of the source, for example $D_{d1d2} = D_{d1} - D_{d2}$.

Consider the first term, $R_{ss}(-E_{d1d2}t + \gamma_d + \tau - D_{d1d2})$, in Equation 3.24. At the assumed value for the differential time delay $\tau = D_{d1d2}$ and relative time companding $\epsilon = E_{d1d2}$, the argument of $R_{ss}[\cdot]$ will become zero and the autocorrelation will reach a maximum value, where other terms will undergo smearing of their correlation peaks, and a large peak will occur in the correlation function. This smearing effect will be shown later in the next chapter. For a single moving source in a multipath environment, the expected value of the time average cross-correlation, $E[\hat{R}_{12}(\tau, \epsilon)]$, over a defined range $\tau$ and $\epsilon$ should have four distinct peaks at value of $D_{d1d2}$, $D_{d1s2}$, $D_{s1d2}$ and $D_{s1s2}$ with corresponding value of relative time companding of $E_{d1d2}$, $E_{d1s2}$, $E_{s1d2}$ and $E_{s1s2}$, respectively. The value of $\tau$ at these distinct peaks are the TDOA estimates for the received signals at the two sensors.

### 3.3 TDOA Estimation for a Moving Source with Interference

Consider the case of a single source with interferer and a pair of sensors with various paths as shown in Figure 3.4. The received signal at each sensor consists of the sum of four waveforms resulting from the direct and surface bounce paths between the source and the sensor and between the interferer and the sensor. The broadband signal, which is generated by the contact source, is denoted $s_C(t)$, where the subscript $C$ is an abbreviation for contact. The broadband signal generated by the interferer is denoted $s_I(t)$, where the subscript $I$ is an abbreviation for interference. The received signals at the two sensors are given by

$$r_1(t) = g_{C1} s_C(t - E_{C1} t - D_{C1}) + g_{C1} s_C(t - E_{C1} t - D_{C1}) +$$
Figure 3.4: A single moving source with interference in a multipath environment
\[ r_2(t) = g_{Cd2sc}(t - E_{Cd2} t - D_{Cd2}) + g_{Cs2sc}(t - E_{Cs2} t - D_{Cs2}) + 
\]
\[ g_{Id2si}(t - E_{Id2} t - D_{Id2}) + g_{Is2si}(t - E_{Is2} t - D_{Is2}) + n_2(t), \]
\text{where}

\[ s_C(t), s_I(t), n_1(t) \text{ and } n_2(t) \text{ are assumed to be stationary, zero mean, uncorrelated and jointly Gaussian random processes,} \]

\[ g_Cdm, g_{Cs m} \text{ and } g_{Id m}, g_{Is m} \text{ are the attenuation coefficients for the direct and surface path of the contact source and the interference source to the } m^{th} \text{ sensor respectively where } m = 1, 2 \text{ and the subscript } d \text{ denotes the direct path and } s \text{ denotes the surface bounce path,} \]

\[ E_{Cd m}, E_{Cs m} \text{ and } E_{Id m}, E_{Is m} \text{ are the time delay rates for the direct and surface path of the contact source and the interference source, respectively,} \]

\[ D_{Cd m}, D_{Cs m} \text{ and } D_{Id m}, D_{Is m} \text{ are the time delays for the direct and surface path of the contact source and interference source, respectively} \]

Using the same analysis as for the case of a single moving source in a multipath environment, the expected value of the time average cross correlation is given by

\[
E\left[ \hat{R}_{12}(\tau, \epsilon) \right] = E\left[ \sum_{n=1}^{N} \frac{1}{T_s} \int_{\gamma_d - \frac{T_s}{2}}^{\gamma_d + \frac{T_s}{2}} r_1(t)r_2(t - \gamma_d \epsilon - \tau) dt \right]
\]

\[
= E\left[ \sum_{n=1}^{N} \frac{1}{T_s} \int_{\gamma_d - \frac{T_s}{2}}^{\gamma_d + \frac{T_s}{2}} \{ g_{Cd1sc}(t - E_{Cd1} t - D_{Cd1}) + g_{Cs1sc}(t - E_{Cs1} t - D_{Cs1}) + g_{Id1si}(t - E_{Id1} t - D_{Id1}) + g_{Is1si}(t - E_{Is1} t - D_{Is1}) + n_1(t) \} dt \right]
\]

\[ + \{ g_{Cd2sc}(t - E_{Cd2} t + \gamma_d \epsilon + \tau - D_{Cd2}) + g_{Cs2sc}(t - E_{Cs2} t + \gamma_d \epsilon + \tau - D_{Cs2}) + g_{Id2si}(t - E_{Id2} t + \gamma_d \epsilon + \tau - D_{Id2}) + g_{Is2si}(t - E_{Is2} t + \gamma_d \epsilon + \tau - D_{Is2}) + n_2(t) \} dt \]

\text{(3.27)}
Since $s_C(t)$, $s_I(t)$, $n_1(t)$ and $n_2(t)$ are assumed zero mean jointly Gaussian

$$
E\left[\hat{R}_{12}(\tau, \epsilon)\right] = \sum_{n=1}^{N} \frac{1}{T_s} \int_{\frac{T_d}{T_s}}^{\frac{T_d}{T_s} + \frac{T_d}{T_s}} E \left[ g_{C_{d1}} g_{C_{d2}} \{s_C(t - E_{C_{d1}}t - D_{C_{d1}}) \\ s_C(t - \gamma_d\epsilon - \tau - E_{C_{d2}}t - D_{C_{d2}}) \} + g_{C_{d1}} g_{C_{d2}} \{s_C(t - E_{C_{d1}}t - D_{C_{d1}}) \\ s_C(t - \gamma_d\epsilon - \tau - E_{C_{d2}}t - D_{C_{d2}}) \} \\
+ g_{C_{d1}} g_{C_{d2}} \{s_C(t - E_{C_{d1}}t - D_{C_{d1}}) \\ s_C(t - \gamma_d\epsilon - \tau - E_{C_{d2}}t - D_{C_{d2}}) \} + g_{C_{d1}} g_{C_{d2}} \{s_C(t - E_{C_{d1}}t - D_{C_{d1}}) \\ s_C(t - \gamma_d\epsilon - \tau - E_{C_{d2}}t - D_{C_{d2}}) \} + g_{I_{d1}} g_{I_{d2}} \{s_I(t - E_{I_{d1}}t - D_{I_{d1}}) \\ s_I(t - \gamma_d\epsilon - \tau - E_{I_{d2}}t - D_{I_{d2}}) \} \\
+ g_{I_{d1}} g_{I_{d2}} \{s_I(t - E_{I_{d1}}t - D_{I_{d1}}) \\ s_I(t - \gamma_d\epsilon - \tau - E_{I_{d2}}t - D_{I_{d2}}) \} + g_{I_{d1}} g_{I_{d2}} \{s_I(t - E_{I_{d1}}t - D_{I_{d1}}) \\ s_I(t - \gamma_d\epsilon - \tau - E_{I_{d2}}t - D_{I_{d2}}) \} + g_{I_{d1}} g_{I_{d2}} \{s_I(t - E_{I_{d1}}t - D_{I_{d1}}) \\ s_I(t - \gamma_d\epsilon - \tau - E_{I_{d2}}t - D_{I_{d2}}) \}} dt
$$

$$
E\left[\hat{R}_{12}(\tau, \epsilon)\right] = \sum_{n=1}^{N} \frac{1}{T_s} \int_{\frac{T_d}{T_s}}^{\frac{T_d}{T_s} + \frac{T_d}{T_s}} E \left[ g_{C_{d1}} g_{C_{d2}} R_{s_C}( - E_{C_{d1}}t + \gamma_d\epsilon + \tau - D_{C_{d1}}t) \\
+ g_{C_{d1}} g_{C_{d2}} R_{s_C}( - E_{C_{d2}}t + \gamma_d\epsilon + \tau - D_{C_{d2}}t) \\
+ g_{C_{d1}} g_{C_{d2}} R_{s_C}( - E_{C_{d1}}t + \gamma_d\epsilon + \tau - D_{C_{d1}}t) \\
+ g_{C_{d1}} g_{C_{d2}} R_{s_C}( - E_{C_{d2}}t + \gamma_d\epsilon + \tau - D_{C_{d2}}t) \\
+ g_{I_{d1}} g_{I_{d2}} R_{s_I}( - E_{I_{d1}}t + \gamma_d\epsilon + \tau - D_{I_{d1}}t) \\
+ g_{I_{d1}} g_{I_{d2}} R_{s_I}( - E_{I_{d2}}t + \gamma_d\epsilon + \tau - D_{I_{d2}}t) \\
+ g_{I_{d1}} g_{I_{d2}} R_{s_I}( - E_{I_{d1}}t + \gamma_d\epsilon + \tau - D_{I_{d1}}t) \\
+ g_{I_{d1}} g_{I_{d2}} R_{s_I}( - E_{I_{d2}}t + \gamma_d\epsilon + \tau - D_{I_{d2}}t) \} dt
$$
\[ +g_{I_{s1}g_{I_{s2}}}R_{s1}(-E_{I_{s1}g_{I_{s2}}}t + \gamma_{d}t + \tau - D_{I_{s1}g_{I_{s2}}}) \] 

where

\( E_{C_{j_{k}}} \) is the relative time companding of the various paths of the Contact source and \( E_{I_{j_{k}}} \) is the relative time companding of the various paths of the Interferer, respectively, \( j = d_{1}, s_{1} \) and \( k = d_{2}, s_{2} \).

\( D_{C_{j_{k}}} \) is the differential time delay for the various paths of the Contact source and \( D_{I_{j_{k}}} \) is the differential time delay for the various paths of the Interferer respectively.

The presence of the interference will add several more peaks in the correlograms since the interference signal is included in the received signal. If the value of \( \tau \) and corresponding value of \( \epsilon \) for the peaks from the interference are different from the value of \( \tau \) and corresponding value of \( \epsilon \) for the peaks from the source then the correlograms should have eight distinct peaks at \( D_{C_{d1}d_{2}}, D_{C_{d1}s_{2}}, D_{C_{s1}s_{2}} \) and \( D_{C_{s1}d_{2}} \) for the source, and \( D_{I_{d1}d_{2}}, D_{I_{d1}s_{2}}, D_{I_{s1}s_{2}} \) and \( D_{I_{s1}d_{2}} \) for the interference. The effect of the compensation term \( \epsilon \) and the use of short time correlation technique can be shown by further deriving the expression for \( E[\hat{R}_{ij}(\tau, \epsilon)] \). First the autocorrelation can be written in the form:

\[ \hat{R}_{c}(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(w)e^{jw\tau} dw \]

Therefore the expected value can be written as:

\[ E[\hat{R}_{ij}(\tau, \epsilon)] = \sum_{n=1}^{N} \frac{1}{T_{s}} \int_{\gamma_{d} - \frac{T_{s}}{2}}^{\gamma_{d} + \frac{T_{s}}{2}} \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[ \left. \hat{S}_{C}(w) \right| \left. g_{C_{d1}g_{C_{d2}}}e^{jw(-E_{C_{d1}d_{2}}t + \gamma_{d}t + \tau - D_{C_{d1}d_{2}})} \right. 
\right. + \left. g_{C_{d1}g_{C_{s2}}}e^{jw(-E_{C_{d1}s_{2}}t + \gamma_{d}t + \tau - D_{C_{d1}s_{2}})} \right. 
\right. + \left. g_{C_{s1}g_{C_{d2}}}e^{jw(-E_{C_{s1}d_{2}}t + \gamma_{d}t + \tau - D_{C_{s1}d_{2}})} \right. 
\right. + \left. g_{C_{s1}g_{C_{s2}}}e^{jw(-E_{C_{s1}s_{2}}t + \gamma_{s}t + \tau - D_{C_{s1}s_{2}})} \right. 
\right. \left. + \left. S_{I}(w) \right| \left. g_{I_{1}g_{I_{d2}}}e^{jw(-E_{I_{1}d_{2}}t + \gamma_{d}t + \tau - D_{I_{1}d_{2}})} \right. \right] 
\]
\[ +g_{1d1}g_{1s2}e^{jw(-E_{1d1}s_{2}+\gamma_{d}t+\tau-D_{1d1}s_{2})} \]
\[ +g_{1s1}g_{1d2}e^{jw(-E_{1s1}d_{1}t+\gamma_{s}t+\tau-D_{1s1}d_{2})} \]
\[ +g_{1s1}g_{1s2}e^{jw(-E_{1s1}s_{2}t+\gamma_{s}t+\tau-D_{1s1}s_{2})}\] 
\[ d\omega \] 

(3.29)

But
\[ \int_{nT_{d}-\frac{T_{d}}{2}}^{nT_{d}+\frac{T_{d}}{2}} e^{jw(-E_{d1}d_{2}t)} dt = \frac{1}{-jwE_{d1}d_{2}} \left[ e^{-jwE_{d1}d_{2}\gamma_{d}} - e^{-jwE_{d1}d_{2}(\gamma_{d}-\frac{T_{d}}{2})} \right] \]
\[ = \frac{-jwE_{d1}d_{2}}{jwE_{d1}d_{2}} \left[ e^{jwE_{d1}d_{2}\gamma_{d}} - e^{-jwE_{d1}d_{2}\gamma_{d}} \right] \]
\[ = T_{s}e^{-jwE_{d1}d_{2}\gamma_{d}} \text{sinc}(wE_{d1}d_{2}T_{s}/2) \] 

(3.30)

The expected value becomes:
\[ E\left[ R_{12}(\tau, \epsilon) \right] = \sum_{n=1}^{N} \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[ S_{C}(w)g_{C1}g_{C2}e^{jw(-E_{C1}d_{2}+\gamma_{d}t+\tau-D_{C1}d_{2})}T_{s}\text{sinc}(wE_{C1}d_{2}T_{s}/2) \right] \]
\[ +g_{C1}g_{C2}e^{jw(-E_{C1}d_{2}\gamma_{d}+\tau-D_{C1}s_{2})}T_{s}\text{sinc}(wE_{C1}d_{2}T_{s}/2) \]
\[ +g_{C1}g_{C2}e^{jw(-E_{C1}d_{2}\gamma_{s}+\tau-D_{C1}s_{2})}T_{s}\text{sinc}(wE_{C1}d_{2}T_{s}/2) \]
\[ +g_{C1}g_{C2}e^{jw(-E_{C1}d_{2}\gamma_{s}+\tau-D_{C1}s_{2})}T_{s}\text{sinc}(wE_{C1}d_{2}T_{s}/2) \]
\[ S_{1}(w)\left[ g_{1d1}g_{1c2}e^{jw(-E_{1d1}d_{2}+\gamma_{d}t+\tau-D_{1d1}d_{2})}T_{s}\text{sinc}(wE_{1d1}d_{2}T_{s}/2) \right] \]
\[ +g_{1d1}g_{1c2}e^{jw(-E_{1d1}d_{2}\gamma_{d}+\tau-D_{1d1}d_{2})}T_{s}\text{sinc}(wE_{1d1}d_{2}T_{s}/2) \]
\[ +g_{1s1}g_{1c2}e^{jw(-E_{1s1}d_{2}\gamma_{d}+\tau-D_{1s1}d_{2})}T_{s}\text{sinc}(wE_{1s1}d_{2}T_{s}/2) \]
\[ +g_{1s1}g_{1c2}e^{jw(-E_{1s1}d_{2}\gamma_{s}+\tau-D_{1s1}d_{2})}T_{s}\text{sinc}(wE_{1s1}d_{2}T_{s}/2) \] 
\[ dw \] 

(3.31)

For real signals \( s_{C}(t) \) and \( s_{1}(t) \), \( R_{S_{C}}(\tau) \) and \( R_{S_{1}}(\tau) \) are real and even. Thus \( S_{C}(w) \) and \( S_{1}(w) \) are real and we can replace
\[ e^{jw(-E_{C1}d_{2}\gamma_{d}+\tau-D_{C1}d_{2})} \]

with
which gives

\[
E[\hat{R}_{12}(\tau, \epsilon)] = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[ S_C(w) \left[ g_{C_{d1}C_{d2}} \text{sinc}(wE_{C_{d1}d2}T_s) \right] 
\right. \\
\left. + \sum_{n=1}^{N} \cos[w(-E_{C_{d1}d2}\gamma_d + \gamma_d \epsilon + \tau - D_{C_{d1}d2})] 
+ g_{C_{d1}C_{d2}} \text{sinc}(wE_{C_{d1}d2}\frac{T_s}{2}) \right] dw (3.32)
\]

The resulting equation is a function of the source and interference signal spectral densities. In the expression for \( E[\hat{R}_{12}(\tau, \epsilon)] \) in Equation 3.32, the cosine term will become maximum (equal 1) if its argument is zero. That is, at the assumed value of \( \tau \) and \( \epsilon \), for example for the path pair \( d1d2 \), \( \tau = D_{d1d2} \) and \( \epsilon = E_{d1d2} \), the relative
time companding among short-time intervals $T_d$ is compensated and a large peak will occur in the correlogram. The term $\text{sinc}[.]$ represents the effect of short-time interval $T_d$ in the correlation. When $T_d$ is small, i.e., the main lobe of the $\text{sinc}[.]$ waveform is broad. This reduces the effect of the side lobes in the integration and the resulting correlation peak is sharp. As $T_d$ becomes larger, i.e., the main lobe of the $\text{sinc}[.]$ waveform will become narrower. This increases the effect of the side lobes and the resulting correlation peak will spread out as a result.

3.4 Summary

The mathematical expressions for the expected value of the accumulative time average cross-correlations of the received signals for all cases are derived using the model of linear varying time delay. This model is developed with the assumption all higher order terms in Equation 3.8 are small and neglected. The relative time companding factor is introduced in the argument of the autocorrelation function due to the source motion. This relative time companding has to be compensated in order to obtained meaningful results from the correlation. The short time correlation technique, with the addition of the interval-in-interval correlation technique, is used in the correlation process to estimate the TDOA. This technique proposes taking short time intervals for correlation and summing the resulting correlograms. This is done to improve the signal-to-noise-ratio of the peak in the correlation function and isolate multiple sources which will be explained in the next chapter.
4. The TDOA Estimation

4.1 Introduction

Time Difference of Arrival (TDOA) estimation is critical in passive source localization. With the assumption of constant sound speed, the TDOA estimate defines a hyperboloid containing possible locations of the source corresponding to that TDOA [5]. The intersection of these hyperboloids, defined by the TDOA estimates, will produce the source location. The accuracy of the TDOA estimates can affect the accuracy of the outcomes of the source localization subsystems.

The problem in the TDOA estimation is due to the source motion, where the received signals are Doppler-scaled at each sensor leading to relative time companding and a loss in signal coherence. The multipath environment introduces more complexity in the existing problem with additional path signals in the received signal at the sensor. The presence of an interferer adds several more peaks in the cross-correlation and autocorrelation which increase the possibility of interfering TDOA peaks in the Passive Ambiguity Surface.

This chapter introduces an algorithm for the TDOA estimation using the Deskewed Short-Time Correlator technique. Another algorithm based on the Select-Correlate-Sum (SCS) technique by K. Jeffrey [9], is also developed. This algorithm is similar to the proposed algorithm and suggested as an alternative to the proposed algorithm. A new technique to improve the resulting TDOA estimates, the Interval-in-Interval correlation technique, is also introduced.
4.2 Short Time Correlation Algorithm

The main focus in this work consists of two moving sources in a multipath environment where only direct and surface reflected paths are considered for simplification. When a pair of sensors are used for the TDOA estimation, a general model for the signal received at the \( m \)th sensor can be given by

\[
r_m(t) = g_{Cdm}s_C(t - D_{Cdm}(t)) + g_{Csm}s_C(t - D_{Csm}(t)) + g_{Irm}s_I(t - D_{Irm}(t)) + g_{Ism}s_I(t - D_{Ism}(t)) + n_m(t),
\]

where \( s_C(t) \) is a random, broadband signal produced by the contact source and \( s_I(t) \) is a random, broadband signal produced by the interfering source. The sensor receives two signals from the contact source, the direct path waveform with a time varying attenuation coefficient, \( g_{Cdm}(t) \), and time varying delay rate \( D_{Cdm}(t) \), and the surface reflected waveform with attenuation and time delay rate \( g_{Csm}(t) \) and \( D_{Csm}(t) \), respectively. The sensor also receives two signals from the interfering source, the direct path waveform with an attenuation coefficient, \( g_{Irm}(t) \), and time delay rate \( D_{Irm}(t) \), and the surface reflected waveform with attenuation and time delay rate \( g_{Ism}(t) \) and \( D_{Ism}(t) \), respectively.

The algorithm proposed here addresses the time scaling problem, as mentioned earlier in Chapter 3, by breaking up the integration interval, \([0, T]\), into short time intervals, \([0, T_d]\), over which the time delay can be assumed constant. These smaller intervals are taken every \( T_d \) seconds where \( NT_d = T \). The time average cross-correlation over each of these smaller intervals can then be summed giving the expression:

\[
\hat{R}_{12}(\tau, D(\gamma_n)) = \sum_{n=1}^{N} \frac{1}{T_d} \int_{\gamma_n - \frac{T_d}{2}}^{\gamma_n + \frac{T_d}{2}} r_1(t)r_2(t - D(\gamma_n)\gamma_n - \tau)dt
\]

where \( \gamma_n = -T/2 + \frac{2n-1}{2}T_d \), \( N \) is odd and \( \beta(\gamma_n) \) is a time companding factor which is approximately constant over the interval \( T_d \). An example of linear time varying can be obtained by setting \( \beta(\gamma_n) = \epsilon \), where \( \epsilon \) is a constant.

The advantage of this algorithm in isolating multiple sources will be investigated for the less complex case of linear time varying delay. The received signal is modelled
using a Taylor series expansion of the time varying delay about $t = 0$ as

$$r_m(t) = g_{Cdm}g_{sC}(t - E_{Cdm}t - D_{Cdm}(t)) + g_{Csm}g_{sC}(t - E_{Csm} - D_{Csm}(t)) +$$

$$g_{Idm}g_{sI}(t - E_{Idm} - D_{Idm}(t)) +$$

$$g_{Ism}g_{sI}(t - E_{Ism} - D_{Ism}(t)) + n_m(t),$$

(4.3)

where $D_{Ckm} (D_{Ikm})$ is the signal transit time at $t = 0$ (time delay) for the various paths $\{k = d, s; d$ - direct path, $s$ - surface bounce path $\}$ from the contact (interferer) to the $m^{th}$ sensor, and $E_{Ckm} (E_{Ikm})$ is the time delay rate at $t = 0$ for the various paths $\{k = d, s; d$ - direct path, $s$ - surface bounce path $\}$ from the contact (interferer) to the $m^{th}$ sensor. Over a short distance, the variation of the attenuation coefficients will be small. Thus, it can be assumed to be constant and takes on their $t = 0$ value. The time delay rate is a function of the velocity component in the direction of the sensors. For example, the time delay rate for the contact along the direct path to sensor $d_1$ is $E_{Cd1} = V_{d1}/c$ where $V_{d1}$ is the velocity component of the source in the direction of sensor $d_1$ and $c$ is the speed of sound underwater.

The general time average cross-correlation of Equation 4.2, can be simplified for linear time varying delay by substituting $\beta(\gamma_n) = \epsilon$ where $\epsilon$ is a constant to give

$$\hat{R}_{12}(\tau, \epsilon) = \sum_{n=1}^{N} \frac{1}{T_s} \int_{\gamma_n - \frac{T_s}{2}}^{\gamma_n + \frac{T_s}{2}} r_1(t)r_2(t) dt$$

(4.4)

Some insight into the short time correlator can be gained from the expected value of Equation 4.4. This can be obtained by substituting the received signal of Equation 4.3 into Equation 4.4 and taking the expected value, which gives

$$E \left[ \hat{R}_{12}(\tau, \epsilon) \right] = \sum_{n=1}^{N} \frac{1}{T_s} \int_{\gamma_n - \frac{T_s}{2}}^{\gamma_n + \frac{T_s}{2}}$$

$$\left[ g_{Cd1}g_{Cd2}R_{sc}(-E_{Cd1}t + \gamma_n\epsilon + \tau - D_{Cd1}t) + g_{Cd1}g_{Cd2}R_{sc}(-E_{Cd1}t + \gamma_n\epsilon + \tau - D_{Cd1}t) + g_{Ca1}g_{Cd2}R_{sc}(-E_{Ca1}t + \gamma_n\epsilon + \tau - D_{Ca1}t) + g_{Ca1}g_{Cd2}R_{sc}(-E_{Ca1}t + \gamma_n\epsilon + \tau - D_{Ca1}t) + g_{Id1}g_{Id2}R_{sI}(-E_{Id1}t + \gamma_n\epsilon + \tau - D_{Id1}t) \right]$$

$$dt$$
where \( D_{Cij} = D_{Ci} - D_{Cj} \), \( E_{Cij} = E_{Ci} - E_{Cj} \); \( i, j = \{d_1, s_1, d_2, s_2\} \), for example \( D_{C1d_2} = D_{C1} - D_{C2} \) and \( E_{C1d_2} = E_{C1} - E_{C2} \), and \( D_{Iij} = D_{Ii} - D_{Ij} \), \( E_{Iij} = E_{Ii} - E_{Ij} \); \( i, j = \{d_1, s_1, d_2, s_2\} \), for example \( D_{Id_1d_2} = D_{Id_1} - D_{Id_2} \) and \( E_{Id_1d_2} = E_{Id_1} - E_{Id_2} \), and \( E_{Cij}, E_{Iij} \) are much smaller than 1 is used to simplify the equation. This equation is the same as Equation 3.28 derived earlier in Section 3.4, Chapter 3.

Figure 4.1 shows a plot of the autocorrelation peak of the first term in Equation 4.5. The location of the peak of \( R_{ss}(-E_{C1d_2}t + \gamma_n \epsilon + \tau - D_{C1d_2}) \) with \( \epsilon = 0 \), indicated by the solid line, is a plot of the equation

\[
t = \frac{1}{E_{C1d_2} \tau - \frac{D_{C1d_2}}{E_{C1d_2}}} (4.6)
\]

The relative time companding factor, \( \epsilon \), is set to zero to show how the autocorrelation varies over time, \( t \). The short time correlation algorithm integrates with respect to
the $t$ variable over $N$ intervals of width $T_s$. Two of these integration intervals are shown in Figure 4.1. In the interval about $i = 0$ (and all other intervals), performing the integration will have the effect of smearing the peak, and this smearing will be proportional to the slope of the line, $\frac{1}{E_{Cd1d2}}$, and the length of the integration interval $T_s$. If there is no movement, $E_{Cd1d2} = 0$ and the line will be vertical resulting in no smearing of the peak, since the correlation function will not be a function of time, $t$. The effect of the relative time companding factor is to shift each short time integral horizontally. If the relative time companding factor estimate is exact, $\epsilon = E_{Cd1d2}$, then the short time intervals will be aligned vertically in Figure 4.1. Thus summing the $N$ short time integrals will not result in any further smearing of the peak. Note that the other terms in Equation 4.5 will undergo smearing of their correlation peaks when the $N$ short time intervals are summed if $\epsilon = E_{Cd1d2}$, and $E_{Cd1d2}$ is different from the other relative time companding factors $E_{Cd1d2}, E_{Cs1d2}, E_{Id1d2}$, $E_{Id1d2}, E_{Id1d2}, E_{Id1d2}$. This effect, which can be used to reduce the interference of other sources, is a major advantage of this algorithm.

4.3 The Interval-in-Interval Correlation Technique

Another technique, the interval-in-interval correlation technique, is introduced. Consider Equation 4.5 for the case of two moving sources. Instead of taking short time intervals right beside each other in the data sequence, a data segment $T_d$ will be taken within $T_d$, for correlation. As a result, longer data sequences are needed for the correlation process. This is shown in Figure 4.2 with some exaggeration for the length of $T_d$ to make the difference of the two sequences more obvious. The variation of the amount of time shifting $\gamma_n \epsilon$, required to compensate for the relative time companding of the received signal, will become greater for the same number of short time intervals used in the correlation. This will not have much effect on the autocorrelation peak of the term with the exact relative time companding factor but it will cause more smearing of the autocorrelation peaks of the terms with different relative
Using interval-in-interval correlation technique

Figure 4.2: Data sequence for correlation

The presence of noise in the received signal has the effect of corrupting the sample sequences and degrading the similarity of the delayed versions of the sound emitted by the source of interest. This affects the results from the correlation, i.e., changes the shape of the correlogram. Most importantly, noise introduces additional
variance into locating the peak of the correlogram main lobe and therefore into estimating the TDOA. The sharpness of the peak, the narrowness of the main lobe, affects the degree to which noise can affect the location of the peak.

An interference will adversely affect the resulting correlation peak for the contact source. The sound signal from both the source and the interferer are presented in the sequences being correlated. The interfering source signal acts like an additional noise source to the received signal for the source of interest. This additional noise will corrupt the received signal and reduce the sharpness of the correlation peaks further.

The effect of the noise and interference is shown Figure 4.3. The sharpness of the main lobe in the correlogram reduces with the addition of noise and interference, which in turn reduces the accuracy of the TDOA estimates. The degree of this effect varies depending on the power level of the noise and interference.

The presence of the interference may also affect the accuracy of the TDOA estimates obtained from the contact source. Since the two sources are independent, the resulting correlogram is like the sum of two correlograms obtained by separately correlating the contact and interferer components. A local peak in the contact diagram could be shifted or changed when combined with the correlogram from the interferer. If the interferer is strong, then the side lobes in the interferer correlogram could be large enough to affect the main lobe of the contact correlogram. Figure 4.4 shows a simulation of the effect of a strong interferer on the contact correlation peak. Both correlograms are produced with the same delay $\tau$ and relative time companding $\epsilon$. The peak of the contact correlogram is slightly decreased and shifted due to side lobes of the interferer correlogram.

4.5 The TDOA Estimation Algorithms
4.5.1 Algorithm I

Algorithm I, which is suggested in this work as an alternative algorithm, is based on the Select-Correlate-Sum (SCS) technique by K. Jeffrey [9] which is further
expanded to include the multipath case. This algorithm proposes compensation of the relative time companding of the received signals by time shifting and then correlating short-time intervals. The resulting short time correlograms will be accumulated to give the time average cross correlation that will be used to estimate the TDOA. In addition to the SCS technique the interval-in-interval correlation technique, which is explained earlier, is used to further reduce the effect of interference.

Figure 4.5 shows a block diagram of the SCS technique with the addition of the interval-in-interval correlation technique. The received signals $r_1(t)$ and $r_2(t)$ pass through a low pass filter. This low pass filter is used to filter the signal based on the assumption that the received signal is a frequency limited broadband signal.

After filtering, $r_2(t)$ is delayed by $K$ different values in a defined range of the differential time delay $\tau$ to give $r_2(t - \tau_k)$, where $k = 1, 2, \ldots, K$. Short time interval $T_d$ of $r_2(t - \tau_k)$ will be time shifted by $M$ different values in a defined range of the relative time companding $\epsilon$ which gives $r_2(t - \tau_k - \epsilon_m \gamma_d)$, where $m = 1, 2, \ldots, M$, $n = 1, 2, \ldots, N$, $N$ is odd, and $\gamma_d = -T/2 + \frac{2n-1}{2}T_d$. A data segment $T_s$ will be taken within each short time interval $T_d$ of the time shifted block output, $r_2(t - \tau_k - \epsilon_m \gamma_d)$, and cross-correlated with a corresponding data segment $T_s$ within $T_d$ of the filtered signal $r_1(t)$. The correlation will be done $N$ times at the cross-correlation block and the resulting short time correlograms will be passed to the storage block.

The complete block diagram of the Algorithm I is shown in Figure 4.6. In the storage block, the resulting short time correlograms will be summed to give the time average cross-correlation

$$\hat{R}_{12}(\tau_k, \epsilon_m) = \sum_{n=1}^{N} \frac{1}{T_s} \int_{\gamma_d - \frac{T_d}{2}}^{\gamma_d + \frac{T_d}{2}} r_1(t) r_2(t - \tau - \gamma_d \epsilon_m) dt$$

and stored appropriately.

The peak detector uses the correlation peaks of stored data in the storage block to form a Passive Ambiguity Surface (PAS). A plot of the PAS formed by simulated results, for the case of a single moving source with interference, using MATLAB is shown in Figure 4.7. The data obtained to form the PAS in this case by using $M = 30$, $K = 70$ and $N = 200$. These numbers for $M$, $K$ and $N$ are chosen with known
source location and velocity. The normal values for $M$, $K$ and $N$ in a simulation are 100, 120 and 200, respectively. Of course, these values can be varied for better accuracy and resolution as needed. There are eight distinct peaks on the PAS, four for the source and four for the interference. Each peak is associated with a correct value of $\tau$, for the differential time delay, and a value of $\beta$, fully compensated for the relative time companding, of a pair of path signals received at the two sensors. The TDOA estimates can be obtained by searching for the position of these distinct peaks on the PAS.

4.5.2 Algorithm II

Algorithm II, which is the proposed algorithm, is based on the Deskewed Short-time correlator (DSTC) technique. The desirable property of this technique is 'correlate first, then compensate'. This technique is similar to the SCS technique where the main difference is in the sequence of execution in the correlation process. Figure 4.8 shows a block diagram of the short time cross-correlator using DSTC technique.

After the received signals $r_1(t)$ and $r_2(t)$ pass through the low pass filter, data segment $T_s$ is taken within uncompensated short time interval $T_d$ of the filtered signals for correlation. The resulting correlograms will be time shifted and summed to produce the time average cross-correlation, whereas in the SCS technique short-time interval $T_d$ are time shifted prior to correlation. The search for the TDOA estimates is the same as in the alternative algorithm. The block diagram of Algorithm II is the same as the one for Algorithm I shown in Figure 4.6.

4.5.3 Comparison of Algorithms

Simulated results using MATLAB in Chapter 6 show that Algorithm II provides similar accuracy compared to that provided by the Algorithm I. The advantage of using the DSTC technique for Algorithm II is that the correlation is done only once for each value of differential time delay which leads to a significant reduction
in computational time. In this case the reduction is $M$ times less correlation, where $M$ is approximately one hundred in the simulation. Comparison of hardware implementation complexity indicates that both algorithms need similar storage, due to the same number of output correlograms, while Algorithm I has the advantage in implementational simplicity.
Figure 4.3: Effect of noise and interference on correlation peak
Figure 4.4: Effect of interference on contact correlation peak
where \( \gamma_d = -T/2 + \frac{n\pi}{2} - \frac{1}{2} T_d \) and \( n = 1, 2, \ldots, N \)., \( N \) is odd.

\[ m = 1, 2, \ldots, M. \]

\[ \epsilon_{m \gamma_d} \]

\[ r_2(t - \tau_k) \]

\[ r_2(t - \tau_k - \epsilon_{m \gamma_d}) \]

---

**Figure 4.5**: Block diagram of the short time cross-correlator for the Algorithm I

---

Short Time Cross-Correlator
Figure 4.6: Block diagram of Algorithm I
Figure 4.7: Plot of the passive ambiguity surface for a single moving source with interference
where $\gamma_d = -\frac{T}{2} + \frac{2n-1}{2}T_d$ and $n = 1, 2, \ldots, N$, $N$ is odd.

$m = 1, 2, \ldots, M$.

\[ \epsilon_m \gamma_d \]

Short Time Cross-Correlator

Figure 4.8: Block diagram of the short time cross-correlator for Algorithm II
5. The TDOA Registration

5.1 Problem Approach

In practise, little would be known about the location and velocity of the source and the interferer. The TDOA estimates are only useful for the source localization if they are correctly registered to the pair of path signals that produce each TDOA estimate. In order to register the TDOA estimates, all information and relationships of the TDOAs extracted from the TDOA estimation have to be used. Consider Figure 3.4, a diagram for the two-source-two-sensor configuration. For a moving source with interference, cross-correlating the received signals at the two sensors results in two sets of TDOA estimates, one for the source of interest and the other for the interference; \( D_{d1d2} \) from direct path to sensor 1 and direct path to sensor 2, \( D_{d1s2} \) from direct path to sensor 1 and surface reflected path to sensor 2, \( D_{s1d2} \) from surface reflected path to sensor 1 and direct path to sensor 2, and \( D_{s1s2} \) from surface reflected path to sensor 1 and surface reflected path to sensor 2. Autocorrelating the received signal at each sensor results in two sets of TDOA estimates, one for the source and one for the interference; \( D_{s1d1} \) from surface reflected path and direct path to sensor 1 and \( D_{s2d2} \) from surface reflected path and direct path to sensor 2.

The relationships among the TDOA estimates of a source obtained from the autocorrelation and cross-correlation of the received signals are shown below

\[
\begin{align*}
D_{s1d1} &= D_{s1d2} - D_{d1d2} \\
D_{s2d2} &= D_{d1d2} - D_{d1s2} \\
D_{s1d1} &= D_{s1s2} - D_{d1s2} \\
D_{s2d2} &= D_{s1d2} - D_{s1s2}
\end{align*}
\]
\[ D_{s1d1} + D_{s2d2} = D_{s1d2} - D_{d1s2} \]  \hspace{1cm} (5.5) \\
\[ D_{s2d2} - D_{s1d1} = D_{d1d2} - D_{s1s2} \]  \hspace{1cm} (5.6) \\
\[ D_{s1d2} + D_{d1d2} = D_{d1d2} + D_{s1s2} \]  \hspace{1cm} (5.7)

Since the source and interferer can not easily be identified from one another while both are moving, the interferer may be considered as another moving source. These above relationships among the TDOAs can be used to classify the TDOAs as belonging to one source or to the other source.

Associating the TDOA estimates to the pairs of path signals that produce each TDOA estimate can be done by using the order of the TDOA estimates detected from a source. Figure 5.1 shows the total search space of the TDOA in a two-sensor configuration from a top view. The TDOA estimates obtained from two sources located symmetrically over the x-axis will be the same; for example two sources located symmetrically over the x-axis in the two quadrants, first and fourth, will have the same TDOA estimates. Thus, only two quadrants of the four quadrants in the total space needs to be considered in the search. Two quadrants, first and second, are chosen as reference for the registration.

In the first quadrant, the resulting TDOA estimates are found to have an as-

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure5_1.png}
\caption{Top view of the search space for a two-sensor configuration}
\end{figure}
ascending order according to the time as shown below

\[ D_{d_1s_2} < D_{d_1d_2} < D_{s_1s_2} < D_{s_1d_2} \]

This ascending order can be confirmed by proving the following inequalities: \( D_{d_1d_2} > D_{d_1s_2} \), \( D_{s_1s_2} > D_{d_1d_2} \) and \( D_{s_1d_2} > D_{s_1s_2} \)

i) \( D_{d_1d_2} > D_{d_1s_2} \)

\[ D_{d_1d_2} - D_{d_1s_2} = D_{s_2d_2} > 0 \]
\[ \Rightarrow D_{d_1d_2} > D_{d_1s_2} \]

ii) \( D_{s_1s_2} > D_{d_1d_2} \)

\[ D_{s_1s_2} - D_{d_1d_2} = D_{s_1d_1} - D_{s_2d_2} \]

In order for \( D_{s_1s_2} > D_{d_1d_2} \) to be true then the difference \( D_{s_1d_1} - D_{s_2d_2} \) must be greater than zero. Consider Figure 5.2 which is a diagram of overlaying the triangle formed by the virtual sensors, \( S_{s_1} \) and \( S_{s_2} \), and the source on the triangle formed by the direct sensors, \( S_{d_1} \) and \( S_{d_2} \), and the source. When the source approaches the dashed line, then \( D_{d_1} \) is equal \( D_{d_2} \) and \( D_{s_1} \) is equal \( D_{s_2} \) or \( D_{s_2d_2} = D_{s_1d_1} = 0 \). Examining the two triangles having the same side \( b \), there is a ratio given by

\[ \frac{l_{d_1}}{l_{s_1}} = \frac{a}{b}, \text{ and} \]
\[ \frac{l_{d_2}}{l_{s_2}} = \frac{a}{b}, \text{ or} \]
\[ \frac{l_{d_1}}{l_{s_1}} = \frac{l_{d_2}}{l_{s_2}} \]

where \( l_i \) is the length of the path and \( i \) indicates the path \( d_1, d_2, s_1 \) and \( s_2 \). The above equation can be written as

\[ \frac{l_{s_1} - l_{d_1}}{l_{s_1}} = \frac{l_{s_2} - l_{d_2}}{l_{s_2}} \]
\[ \frac{l_{s_1} - l_{d_1}}{l_{s_1}} = \frac{l_{s_2} - l_{d_2}}{l_{s_2}} \]

In the first quadrant \( l_{s_1} \) is greater than \( l_{s_2} \) thus \( l_{s_1} - l_{d_1} \) is greater than \( l_{s_2} - l_{d_2} \). Dividing both sides of the inequality \( l_{s_1} - l_{d_1} > l_{s_2} - l_{d_2} \) by \( c \), the speed of sound
underwater, gives
\[ D_{s1d1} > D_{s2d2} \]
from which it follows
\[ D_{s1s2} > D_{d1d2} \]

iii) \( D_{s1d2} > D_{s1s2} \)

\[ D_{s1d2} - D_{s1s2} = D_{s2d2} > 0 \]

which follows
\[ D_{s1d2} > D_{s1s2} \]
\[ D_{s1d2} > D_{s1s2}, D_{s1s2} > D_{d1d2} \]
and \( D_{d1d2} > D_{d1s2} \) can be rewritten as
\[ D_{d1s2} < D_{d1d2} < D_{s1s2} < D_{s1d2} \]  \hspace{1cm} (5.8)

In the second quadrant, due to symmetry the resulting TDOA estimates can be found in a similar way to have an ascending order
\[ D_{d2s1} < D_{d2d1} < D_{s2s1} < D_{s2d1} \]

which can be rewritten as
\[ -D_{d2s1} > -D_{d2d1} > -D_{s2s1} > -D_{s2d1} \]

since
\[ D_{s1d2} = -D_{d2s1} \]
\[ D_{s1s2} = -D_{s2s1} \]
\[ D_{d1d2} = -D_{d2d1} \]
\[ D_{d1s2} = -D_{s2d1} \]

The order of the TDOA estimates obtained in the second quadrant is given by
\[ D_{d1s2} < D_{s1s2} < D_{d1d2} < D_{s1d2} \]  \hspace{1cm} (5.9)

In the relationship for the cross-correlation TDOA estimates \( D_{s1d2} \) has the largest algebraic value and \( D_{d1s2} \) has the smallest algebraic value regardless of location of the source. From the geometric source-sensor configuration as shown in
Virtual and direct sensor $S_{s2}$ and $S_{d2}$

Virtual and direct sensor $S_{s1}$ and $S_{d1}$

Figure 5.2: Overlaying direct paths on surface paths
Figure 2.2, in the first quadrant $D_s_1$ is greater than $D_s_2$ and $D_d_1$ is greater than $D_d_2$ which results in positive sign for $D_{s1s2}$ and $D_{d1d2}$. In the second quadrant $D_s_2$ is greater than $D_s_1$ and $D_d_2$ is greater than $D_d_1$ which results in negative sign for $D_{s1s2}$ and $D_{d1d2}$. From the sign of $D_{s1s2}$ and $D_{d1d2}$, the source can be located as in the one half, containing first or fourth quadrants, or the other half, containing second or third quadrant, of the total search space.

### 5.2 The Registration Algorithm

The registration process includes two main steps, classifying and associating the TDOAs. Since the main focus of this work is the problem of two different sources moving on straight line paths past two receivers, it is assumed there are a maximum of two sources present in the source localization. Consider a simple system block diagram as shown in Figure 5.3. The first step in the classification is identifying the

![Figure 5.3: Block diagram for the TDOA estimates registration](image-url)
TDOA estimates. This task can be done using the relationships between the TDOA estimates from the cross-correlation and the autocorrelation which are given by

\[
D_{s1d1} + D_{s2d2} = D_{s1d2} - D_{d1s2}, \quad \text{and} \\
D_{s2d2} - D_{s1d1} = D_{d1d2} - D_{s1s2}
\]

These relationships will help identifying the two pair of TDOA estimates \(\{D_{s1d2}, D_{d1s2}\}\) and \(\{D_{d1d2}, D_{s1s2}\}\). Since \(D_{s1d2}\) is always greater than \(D_{d1s2}\), \(D_{s1d2}\) and \(D_{d1s2}\) can be identified by comparing them. \(D_{d1d2}\) and \(D_{s1s2}\) can be identified from each other by checking the sign, i.e. if both have positive sign then \(D_{d1d2}\) is greater than \(D_{s1s2}\) or if both have negative sign then \(D_{d1d2}\) is smaller than \(D_{s1s2}\). These two pairs of TDOA estimates can be classified to the corresponding source using the relationship

\[
D_{s1d2} + D_{d1s2} = D_{d1d2} + D_{s1s2}
\]

In classifying the TDOA estimates, it is important that the possibility of missing peaks are being considered. The missing peaks may be calculated from available peaks detected using the following relationships

\[
D_{s1d1} = D_{s1d2} - D_{d1d2} \\
D_{s2d2} = D_{d1d2} - D_{d1s2} \\
D_{s1d1} = D_{s1s2} - D_{d1s2} \\
D_{s2d2} = D_{s1d2} - D_{s1s2}
\]

Association of the TDOA estimates becomes simple after classifying the TDOA estimates to the corresponding source. First check the sign of the TDOA estimates \(D_{d1d2}\) and \(D_{s1s2}\) to determine which half of the total space the source is located. Then use the corresponding order of the TDOA estimates for the registration. If the source is in the one half containing the first and fourth quadrants, i.e. \(D_{d1d2}\) and \(D_{s1s2}\) have positive sign, then the following order is used

\[
D_{s1d2} > D_{s1s2} > D_{d1d2} > D_{d1s2}
\]

If the source is in the one half containing the second and third quadrants, i.e. \(D_{d1d2}\)
and $D_{s1s2}$ have negative sign, then the following order is used

$$D_{s1d2} > D_{d1d2} > D_{s1s2} > D_{d1s2}$$

The TDOA estimates are associated by the algebraic value in time that is, for the source located in the first quadrant, the one with the largest value is $D_{s1d2}$, the second largest is $D_{s1s2}$, the third largest is $D_{d1d2}$ and the smallest is $D_{d1s2}$. If the source is located in the second quadrant then the one with the largest value is $D_{s1d2}$, the second largest is $D_{d1d2}$, the third largest is $D_{s1s2}$ and the smallest is $D_{d1s2}$.

The presence of false peaks due to excessive noise may produce false results in the registration of the TDOA estimates. Each TDOA estimate is associated with a corresponding relative time companding where false peak due to noise is not. By observing the changing rate of the registered TDOA estimates, correct results can be easily detected.
6. Simulation and Results Comparison

6.1 Signal Simulation

6.1.1 Model Implementation

Figure 6.1 shows the block diagram for the signal generator, which is an implementation of the source and transmission model shown in Figure 2.5, for a single source. The random number generators at the top of Figure 6.1 produce three independent white Gaussian sequences with zero mean and the appropriate variance. This is done by generating three independent random variables. All the sequences are filtered with digital 8th order low pass Butterworth filters having 3 dB points at $\frac{1}{10}$ of the sampling frequency, i.e., $\frac{1}{10T_s} = \frac{f_s}{10}$.

The resulting filtered sequences, $n_1[nT_s]$ and $n_2[nT_s]$ represent the additive Gaussian noise for receiver 1 and receiver 2, respectively. The filtered sequence $s[nT_s]$ represents the signal produced by the source. Because the source is moving, the arrival times of the propagated signal points can no longer be assumed to be uniformly spaced.

To create a uniformly sampled received signal sequence, the received representation is resampled at uniform periods. Data points are resampled by interpolation using a sinc function based on the following formula taken from Oppenheim and Schafer [7]. The value of a signal at any time, $x_a(t)$, can be calculated from available samples, $x_a(kT)$

$$x_a(t) = \sum_{k=-\infty}^{\infty} c_k \phi_k(t)$$  \hspace{1cm} (6.1)

where

$$c_k = x_a(kT),$$
Figure 6.1: Block diagram of the implementation of the ocean transmission model in software
and

\[ \phi_k(t) = \frac{\sin((\pi/T)(t - kT))}{(\pi/T)(t - kT)} \]  \hspace{1cm} (6.2)

This technique was selected for its superior accuracy over the linear interpolation method. This resampling process is done by the interpolation block for each of the paths from the source to the receivers. These path signals are then appropriately delayed. The delays for these paths are calculated assuming a constant sound speed of \( c = 1500 \) meters/second and a horizontal reflection boundary at the surface.

After interpolating and delaying, each path signal is multiplied by a frequency independent attenuation constant \([i]_i\), \( g_i \) where \( i \) takes on values of \( d1, d2, s1 \) and \( s2 \), given by

\[ g_i^2 = \frac{r_s}{l_i^2} 10^{-1/10(a_c l_i + a_s)} \]  \hspace{1cm} (6.3)

where \( l_i \) is the path length in meters, defined in Section 2.1.2, \( a_c \) is the absorption coefficient in \( dB/meter \), \( a_c = 0.2187 \times 10^{-3} dB/meter \), \( a_s \) is the surface attenuation coefficient in \( dB \) (assumed to be zero for simplicity) and \( r_s = -1 \) for a surface reflection path and \( r_s = 1 \) for a direct path. Noise will be added to the sum of the resulting path signals at each receiver. Note that the noise will be generated at a constant power level. The received signals are generated with the assumptions that the source signal frequency is between 0 and 400 Hz and sampling frequency is 8000 samples/second in this simulation.

### 6.1.2 Simulation Procedure

Data is simulated first for the source in one second intervals. This is done for the benefit of keeping the memory, required to run the program, small. During each interval from the beginning of the trial, the spatial position of the source is calculated for each instant. A source motion track, consisting of a starting location, speed and linear direction is specified. Using the instantaneous positions and the known locations of the two receivers, a discrete channel vector is computed which includes the
travel time and propagation loss for each time instant during the one second period. Finally a signal, low pass band limited to 400 Hz, is simulated for the source at the same instants. Channel vectors are created only for the direct propagation paths.

A representation (attenuated value and arrival time) of the signal arriving at each of the receivers is computed by processing each generated signal point with the corresponding travel time and loss from the channel vector. The exact time of arrival of the first data point in each one second interval is saved for later use in verifying implementation of the algorithms. The newly resampled source signal at each receiver is saved as a separate data file. Signal simulation then continues with the next interval.

When all intervals for the specified observation period are simulated for the source, the simulation is repeated for the interfering signal, presumably using some different parameters of starting location, direction, speed and signal power.

Finally, coincident values for source and interference at each sensor are summed second by second appropriately and then combined into one sequence to give the received signal. Incoherent noise is simulated and filtered using the same type of signal filter which will be added to the received signal. The software code for implementing the signal simulation in MATLAB is included in Appendix A.1.

6.2 TDOA Estimation Simulation

The simulation implementing Algorithm II is done using MATLAB programming. Observation data sequences of both filtered received signals $r_1(n)$ and $r_2(n)$ are taken, where $r_2(n)$ is delayed by a value of differential time delays $\tau$. Both sensor 1 sequence $r_1(n)$ and sensor 2 sequence $r_2(n)$ are partitioned into $N$ segments of length $T_d$, where $N$ is equal to 200 in the simulation. A smaller data segment $T_s$ is taken from both data sequences every $T_d$ seconds for correlation. The resulting correlograms are time shifted and summed for different values of $\epsilon$ in a defined range of relative time companding factors. This process will be repeated for different assumed values of $\tau$ in a defined range of differential time delays. The simulation implementing the alter-
native algorithm is done similarly except time shifting short time intervals $T_d$ is done before correlation. The remaining steps in the simulation procedure are identical in implementing both algorithms.

At the correct value of the differential time delay and time shifting rate (differential time delay rate), that is, where $r_2(n)$ is compensated, accumulated correlograms will produce a large peak. The signal to noise ratio (SNR) of the correlation peak in the correlograms improves with increasing number of accumulated segment correlograms. This is shown in Figure 6.2.

Simulated results of correlation peaks obtained from implementing both algorithms are similar. Figure 6.3 and Figure 6.4 show the resulting correlation peaks in implementing Algorithm I and II. The correlation peaks obtained by using Algorithm I are slightly better in sharpness than that obtained by using the Algorithm II.

The peaks of resulting accumulated correlograms from the correlation process are stored in a two dimensional matrix. A graphical representation of this matrix forms a surface known as Passive Ambiguity Surface (PAS). The positions of the peaks in the correlograms might not coincide with the step size of $\tau$. Therefore, interpolation is necessary to find the exact position of the peak.

A method of interpolation [8] is used here which gives relatively accurate result. In the correlograms a parabola is fit to the peak value and the adjacent points. The location of the apex of the parabola is assumed to be the location of the peak of the correlogram. The estimate of differential time delay, denotes $\hat{D}$, is given by

$$\hat{D} = \frac{\hat{R}_{12}[p + 1] - \hat{R}_{12}[p - 1]}{2(-\hat{R}_{12}[p + 1] + 2\hat{R}_{12}[p] - \hat{R}_{12}[p - 1])} + p, \quad (6.4)$$

where $p$ is the integer part of $\hat{R}_{12}[pT_{sp}]$, and $\hat{R}_{12}[pT_{sp}]$ is the member of the sequence $\hat{R}_{12}[mT_{sp}]$ that is the largest (most negative in the case of surface-direct path pair signals) in the vicinity of the delay of interest and $T_{sp}$ here is the sampling interval. The interpolation is done prior to forming the PAS.

The PAS for two moving sources is shown in Figure 4.7. The units on the horizontal axes correspond to the value of step size where the zero value indicates the
Figure 6.2: Improvement in the SNR of the correlation peak with increased number of accumulated segment correlograms
Figure 6.3: Correlation peaks obtained by using Algorithm I
Sum of 200 segment correlations

Figure 6.4: Correlation peaks obtained by using Algorithm II
start value for $\tau$ and $\epsilon$. As shown in Figure 4.7 the step size for $\epsilon$ is 0.0002 and the step size for $\tau$ is 3.125 millisecond. The resolution of the result increases with smaller step size. The positions of the distinct peaks on this surface correspond to the TDOA estimates of the moving source.

The search for the TDOA estimates is implemented using MATLAB, which starts with finding the position of the peak on each of the curves along the $\epsilon$-axis on the PAS. This is done to confine the positions of the distinct peaks from a two dimensional surface to one dimensional curve. A threshold value is set above the level of background noise on the PAS to reduce the possibility of detecting false peaks. The resulting coordinates of the points with magnitude greater than the threshold value are stored in a three column matrix where corresponding value of $\epsilon, \tau$ and peak magnitude are placed in the first, second and third column, respectively.

The data array stored in the magnitude column will form a curve with positive and negative parabolas caused by positive and negative peaks. The values of three consecutive points of the data array will be compared where the point, at which the slope changes sign, is the apex of the parabola. The positions of the apexes of these parabolas are the TDOA estimates of the sources.

The resulting TDOA estimates will be stored in a different three column matrix. The TDOA estimates are arranged in an ascending order of their arithmetic value in time and stored in the second column of the matrix with corresponding relative time companding stored in the first column. The third column will be reserved for the row number. This arrangement of data storing is done in preparing for the registration. The software code for implementing the TDOA estimation algorithm in MATLAB is included in Appendix A.2.

### 6.3 The Registration Simulation

The presence of TDOA estimates from the autocorrelation is essential in the registration of the TDOA estimates obtained from the cross-correlation. Implement-
The first model, with its block diagrams shown in Figure 6.5, is used in the case where the TDOA estimates obtained from the autocorrelation, \( D_{s1d1} \) and \( D_{s2d2} \), are present. The sum and difference of \( D_{s1d1} \) and \( D_{s2d2} \) are calculated in blocks numbered 1 and 3, respectively. In block number 2 subtraction of the TDOA estimates obtained
from the cross-correlation are performed pairwise. The resulting differences will be compared with \( D_{s2d2} + D_{s1d1} \) and \( D_{s2d2} - D_{s1d1} \) in the comparison block (number 4) using the relationships

\[
D_{s2d2} + D_{s1d1} = D_{s1d2} - D_{d1d2}, \text{ and }
\]
\[
D_{s2d2} - D_{s1d1} = D_{d1d2} - D_{s1d2}
\]

From the comparison results these pairs of TDOA estimates can be identified as either \{\( D_{s1d2}, D_{d1d2} \)\} or \{\( D_{d1d2}, D_{s1d2} \)\}. These pairs of \( D_{s1d2}, D_{d1d2} \) and \( D_{d1d2}, D_{s1d2} \) are then passed to the block numbered 5 for classification as belonging to the same source using the relationship

\[
D_{s1d2} + D_{d1d2} = D_{d1d2} + D_{s1d2}
\]  \hspace{1cm} (6.5)

The pairs that meet this condition will be registered in the registration block. First the signs of \{\( D_{d1d2}, D_{s1d2} \)\} are checked to determined which half of the space the source is located, i.e., in the one half, containing the first and fourth quadrants, for positive sign and in the other half, containing the second and third quadrants, for negative sign as shown in Figure 5.1. Then the appropriate ascending order of the TDOA estimates, Equation 5.8 or 5.9, will be used to register the TDOA estimates to the pair of path signals that produce each TDOA estimate. This is done by comparing the values of the TDOA estimates with each other.

If there are pairs of \{\( D_{s1d2}, D_{d1d2} \)\} and \{\( D_{d1d2}, D_{s1d2} \)\}, that do not meet the condition in Equation 6.5, which means that either the matching pairs \{\( D_{s1d2}, D_{d1d2} \)\} or \{\( D_{d1d2}, D_{s1d2} \)\} are missing, then these missing TDOA estimates can be calculated using the relationships

\[
D_{s1d1} = D_{s1d2} - D_{d1d2}
\]
\[
D_{s2d2} = D_{d1d2} - D_{d1d2}
\]
\[
D_{s1d1} = D_{s1d2} - D_{d1d2}
\]
\[
D_{s2d2} = D_{s1d2} - D_{s1d2}
\]
The pairs with the matching pairs, calculated in block number 6, are then passed to the registration block to be registered. The results, due to missing TDOA estimates, are less reliable than the results obtained where the pairs meet the condition in Equation 6.5.

The second model, with its block diagram shown in Figure 6.6, is used when some of the TDOA estimates obtained from autocorrelation are missing. The first block in the second model has the same function as the block number 2 in the first model. The resulting pairs from the first block are passed to block number 2 and 3 for identification. In the block number 2, the pairs are identified using relationships

\[ D_{s1d1} = D_{s1d2} - D_{d1d2} \]
\[ D_{s1d1} = D_{s1s2} - D_{d1s2} \]

where the pairs could be \( \{D_{s1d2}, D_{s1d2}\} \) or \( \{D_{s1s2}, D_{d1s2}\} \). In the block number 3, the pairs are identified using relationships

\[ D_{s2d2} = D_{d1d2} - D_{d1s2} \]
\[ D_{s2d2} = D_{s1d2} - D_{d1s2} \]

where the pairs could be \( \{D_{d1d2}, D_{d1s2}\} \) or \( \{D_{s1d2}, D_{s1s2}\} \).

The resulting pairs from both block number 2 and 3 are then passed to block number 4 for classification. The steps from block numbered 4 to 6 in the second model is identical to the steps from block number 5 to 7 in the first model. The difference for both models here is the output results. When the second model is used, more possible results for one source may be expected where only one is correct. As mentioned earlier, the best results are obtained when all the TDOA estimates from the autocorrelation and cross-correlation are present.

The simulation implementing the registration algorithm is done using MATLAB programming. The simulation can be divided into two parts: classification and registration. The two models with the block diagrams as shown in Figure 6.5 and 6.6 are used to cover all cases which takes into account the possibility of missing peaks.
Substraction of TDOA estimates from X-corr. in pair.

Identification using
\[ s_1d_1 = s_1d_2 - d_1d_2 \]
\[ s_1d_1 = s_1s_2 - d_1s_2 \]
\{s_1d_2, d_1d_2\}
\{s_1s_2, d_1s_2\}

Identification using
\[ s_2d_2 = s_1d_2 - s_1s_2 \]
\[ s_2d_2 = d_1d_2 - d_1s_2 \]
\{s_1d_2, s_1s_2\}
\{d_1d_2, d_1s_2\}

Classification of TDOA estimates to one source

Calculation of missing TDOA estimates

Registration of TDOA estimates

Possible Registered TDOA estimates

Figure 6.6: Block diagram for the second model used in the implementation of the registration algorithm
(TDOA estimates). The classification of the TDOA estimates as belonging to one source for all possible cases are done as following

Case 1: All TDOA estimates from the autocorrelation are present

In this case combinations of the autocorrelation are used to produce more reliable results. Possible pairs of differential time delays $D_{s1d2}$, $D_{d1s2}$ and $D_{d1d2}$, $D_{s1s2}$ can be identified and grouped correspondingly with the associated TDOA estimates $D_{s1d1}$ and $D_{s2d2}$ from the autocorrelation if they satisfy one of the following relationships

\[
D_{s1d1} + D_{s2d2} = D_{s1d2} - D_{d1s2}
\]
\[
D_{s2d2} - D_{s1d1} = D_{d1d2} - D_{s1s2}
\]

a) In an ideal situation there would be two pairs of TDOA estimates $D_{s1d2}$, $D_{d1s2}$ and $D_{d1d2}$, $D_{s1s2}$ expected where they can be classified as belonging to the same source by the relationship

\[
D_{s1d2} + D_{d1s2} = D_{d1d2} + D_{s1s2}
\]

b) If either one of the two pairs $D_{s1d2}$, $D_{d1s2}$ and $D_{d1d2}$, $D_{s1s2}$ are missing, they can be found as follows

- If $D_{d1d2}$ and $D_{s1s2}$ are present then $D_{s1d2}$ and $D_{d1s2}$ can be calculated as follow

\[
D_{s1d2} = D_{d1d2} + D_{s1d1}, \text{ and}
\]
\[
D_{d1s2} = D_{s1s2} - D_{s1d1}
\]

or

\[
D_{s1d2} = D_{s1s2} + D_{s2d2}, \text{ and}
\]
\[
D_{d1s2} = D_{d1d2} - D_{s2d2}
\]

- If $D_{s1d2}$ and $D_{d1s2}$ are present then $D_{d1d2}$ and $D_{s1s2}$ can be calculated as follow

\[
D_{d1d2} = D_{s1d2} - D_{s1d1}, \text{ and}
\]
\[
D_{s1s2} = D_{d1s2} + D_{s1d1}
\]
or

\[ D_{d1d2} = D_{d1s2} + D_{s2d2}, \text{ and} \]
\[ D_{s1s2} = D_{s1d2} - D_{s2d2} \]

c) If the TDOA estimates of one source can be classified and the two TDOA estimates, one of the two \( D_{s1d2} \) and \( D_{d1s2} \) and one of the two \( D_{d1d2} \) and \( D_{s1s2} \), of the other source are missing, then the classification of the TDOA estimates for that source will result in two possible estimates

i) If the present two TDOA estimates could be either \( D_{s1d2} \) and \( D_{d1d2} \) or \( D_{s1s2} \) and \( D_{d1s2} \) resulting from the relationship

\[ D_{s1d1} = D_{s1d2} - D_{d1d2} = D_{s1s2} - D_{d1s2} \]

Two possible classifications of the TDOA estimates for the source exist where only one is true

- If the present TDOA estimates are assumed to be \( D_{s1d2} \) and \( D_{d1d2} \), then the remaining TDOA estimates \( D_{d1s2} \) and \( D_{s1s2} \) can be calculated by

\[ D_{s1s2} = D_{s1d2} - D_{s2d2}, \text{ and} \]
\[ D_{d1s2} = D_{d1d2} - D_{s2d2} \]

- If the present TDOA estimates are assumed to be \( D_{s1s2} \) and \( D_{d1s2} \), then the remaining TDOA estimates \( D_{d1d2} \) and \( D_{s1d2} \) can be calculated by

\[ D_{d1d2} = D_{d1s2} + D_{s2d2}, \text{ and} \]
\[ D_{s1d2} = D_{s1s2} + D_{s2d2} \]

ii) If the present two TDOA estimates could be either \( D_{s1d2} \) and \( D_{s1s2} \) or \( D_{d1d2} \) and \( D_{d1s2} \) resulting from the relationship

\[ D_{s2d2} = D_{s1d2} - D_{s1s2} = D_{d1d2} - D_{d1s2} \]

Two possible classifications of the TDOA estimates for the source where only one is true

- If the present two TDOA estimates are assumed to be \( D_{s1d2} \) and \( D_{s1s2} \), then
the remaining TDOA estimates $D_{d1d2}$ and $D_{d1s2}$ can be calculated by

\[ D_{d1d2} = D_{s1d2} - D_{s1d1}, \quad \text{and} \]
\[ D_{d1s2} = D_{s1s2} - D_{s1d1} \]

- If the present two TDOA estimates are assumed to be $D_{d1d2}$ and $D_{d1s2}$, then the remaining TDOA estimates $D_{s1d2}$ and $D_{s1s2}$ can be calculated by

\[ D_{s1d2} = D_{d1d2} + D_{s1d1}, \quad \text{and} \]
\[ D_{s1s2} = D_{d1s2} + D_{s1d1} \]

d) If none of the two sources can be classified and two TDOAs, one of the two $D_{s1d2}$ and $D_{d1s2}$ and one of the two $D_{d1d2}$ and $D_{s1s2}$, for each source are missing then the classification of the TDOA estimates for each source will result in four possible estimates due to unclassified TDOA estimates in the autocorrelation result.

i) The present two TDOA estimates could be either $D_{s1d2}$ and $D_{d1d2}$ or $D_{s1s2}$ and $D_{d1s2}$ resulting from the relationship

\[ D_{s1d1} = D_{s1d2} - D_{d1d2} = D_{s1s2} - D_{d1s2} \]

Four possible classifications of the TDOA estimates for the source where only one is true

- If the present two TDOA estimates are assumed to be $D_{s1d2}$ and $D_{d1d2}$, then the remaining TDOA estimates $D_{d1s2}$ and $D_{s1s2}$ can be calculated by

\[ D_{s1s2} = D_{s1d2} - D_{s2d2}, \quad \text{and} \]
\[ D_{d1s2} = D_{d1d2} - D_{s2d2} \]

- If the present two TDOA estimates are assumed to be $D_{s1s2}$ and $D_{d1s2}$, then the remaining TDOA estimates $D_{d1d2}$ and $D_{s1d2}$ can be calculated by

\[ D_{d1d2} = D_{d1s2} + D_{s2d2}, \quad \text{and} \]
\[ D_{s1d2} = D_{s1s2} + D_{s2d2} \]
ii) The present two TDOA estimates could be either $D_{s1d2}$ and $D_{s1s2}$ or $D_{d1d2}$ and $D_{d1s2}$ resulting from the relationship

$$D_{s2d2} = D_{s1d2} - D_{s1s2} = D_{d1d2} - D_{d1s2}$$

Four possible classifications of the TDOA estimates for the source where only one is true

- If the present two TDOA estimates are assumed to be $D_{s1d2}$ and $D_{s1s2}$, then the remaining two TDOA estimates $D_{d1d2}$ and $D_{d1s2}$ can be calculated by

$$D_{d1d2} = D_{s1d2} - D_{s1d1}, \text{ and}$$

$$D_{d1s2} = D_{s1s2} - D_{s1d1}$$

- If the present two TDOA estimates are assumed to be $D_{d1d2}$ and $D_{d1s2}$, then the remaining TDOA estimates $D_{s1d2}$ and $D_{s1s2}$ can be calculated by

$$D_{s1d2} = D_{d1d2} + D_{s1d1}, \text{ and}$$

$$D_{s1s2} = D_{d1s2} + D_{s1d1}$$

When three out of four TDOA estimates or all four TDOA estimates of a source are missing then the classification of the TDOA estimates for that source is not possible.

**Case 2: If Only the TDOA estimates from the autocorrelation at one receiver are present**

a) If only the TDOA estimates from the autocorrelation at receiver 1 are present

a1) and only one TDOA estimates from the autocorrelation at receiver 2 is present

In this case, first the following relationships are used to classify the source associated with the TDOA from the autocorrelation at receiver 2.

$$D_{s1d1} + D_{s2d2} = D_{s1d2} - D_{d1s2}$$
and

\[ D_{s2d2} - D_{s1d1} = D_{d1d2} - D_{s1s2} \]

i) If one of the two pairs TDOA estimates \( D_{s1d2}, D_{d1s2} \) and \( D_{d1d2}, D_{s1s2} \), or both meeting this condition are present, then those TDOA estimates are classified to that source.

First use the relationship

\[ D_{s1d1} = D_{s1d2} - D_{d1d2} = D_{s1s2} - D_{d1s2} \]

and then the relationship

\[ D_{d1d2} + D_{s1s2} = D_{s1d2} + D_{d1s2} \]

to classify the TDOAs to the second source

- If one of the two pairs of TDOA estimates \( D_{s1d2}, D_{d1d2} \) and \( D_{s1s2}, D_{d1s2} \) for one source is missing, then classification of the TDOA estimates for that source is not possible

ii) If none of the pairs of TDOA estimates \( D_{s1d2}, D_{d1s2} \) and \( D_{d1d2}, D_{s1s2} \) can be found meeting the condition, then use step c) and d) of case 1 for the classification.

Due to the missing TDOA estimates (one from the autocorrelation and one from the cross-correlation), only one source might be classified where a few possible results can be obtained with one correct answer.

a2) and no autocorrelation TDOA at receiver 2 is present.

First use the relationship

\[ D_{s1d1} = D_{s1d2} - D_{d1d2} = D_{s1s2} - D_{d1s2} \]

and then the relationship

\[ D_{d1d2} + D_{s1s2} = D_{s1d2} + D_{d1s2} \]

to classify the TDOA estimates to the sources

- If one of the two pairs of TDOA estimates \( D_{s1d2}, D_{d1d2} \) and \( D_{s1s2}, D_{d1s2} \) or both are missing for one source, then classification for that source is not possible.

b) If only all TDOA estimates from the autocorrelation at receiver 2 are present

Repeat part a) but uses
\[ D_{s2d2} = D_{s1d2} - D_{s1s2} = D_{d1d2} - D_{d1s2} \]

instead of
\[ D_{s1d1} = D_{s2d2} - D_{d1d2} = D_{s1s2} - D_{d1s2} \]

**Case 3:** Only one TDOA estimate from the autocorrelation at each receiver is present

First use the relationships
\[ D_{s1d1} + D_{s2d2} = D_{s1d2} - D_{d1s2} \]

and
\[ D_{s2d2} - D_{s1d1} = D_{d1d2} - D_{s1s2} \]

to check if the TDOA estimates from the autocorrelation are of the same source.

a) If the TDOA estimates from the autocorrelation are of the same source then the TDOAs for that source can be classified

b) If the TDOA estimates for the autocorrelation are of different sources then use
\[ D_{s1d1} = D_{s2d2} - D_{d1d2} = D_{s1s2} - D_{d1s2} \]

for the TDOA estimate from the autocorrelation at receiver 1 and
\[ D_{s2d2} = D_{s1d2} - D_{s1s2} = D_{d1d2} - D_{d1s2} \]

for the TDOA estimate from the autocorrelation at receiver 2 Then use the relationship
\[ D_{d1d2} + D_{s1s2} = D_{s1d2} + D_{d1s2} \]

to classify the TDOA estimates of the cross-correlation to the source.

i) For the TDOA estimates from the cross-correlation of the source associated with the TDOA estimate from the autocorrelation at receiver 1.

- If both pairs of TDOA estimates \( D_{s1d2}, D_{d1d2} \) and \( D_{s1s2}, D_{d1s2} \) are found, then one result for the classification for the TDOAs corresponding to that source is expected.

- If only one pair or none of the two pairs of TDOA estimates \( D_{s1d2}, D_{d1d2} \) and \( D_{s1s2}, D_{d1s2} \) is found then the classification for the corresponding source is not possible.

ii) For the TDOA estimates from the cross-correlation of the source associated
with the TDOA estimate from the autocorrelation at receiver 2.

- If both pairs of TDOA estimates $D_{s1d2}, D_{s1s2}$ and $D_{d1d2}, D_{d1s2}$ are found, then one result for the classification for the TDOA estimates corresponding to that source is expected.

- If only one pair or none of the two pairs of the TDOA estimates $D_{s1d2}, D_{s1s2}$ and $D_{d1d2}, D_{d1s2}$ is found then the classification of the TDOA estimates for the corresponding source is not possible.

**Case 4:** Only one TDOA estimate from the autocorrelation at one receiver is present

Use the relationship

$$D_{s1d1} = D_{s1d2} - D_{d1d2} = D_{s1s2} - D_{d1s2}$$

if the TDOA estimate from the autocorrelation at receiver 1 is present or

$$D_{s2d2} = D_{s1d2} - D_{s1s2} = D_{d1d2} - D_{d1s2}$$

for the TDOA estimate from the autocorrelation at receiver 2 is present. Then use the relationship

$$D_{d1d2} + D_{s1s2} = D_{s1d2} + D_{d1s2}$$

to classify the TDOA estimates from the cross-correlation to the source.

The result expected is similar to part b) i) if the TDOA estimate from the autocorrelation at receiver 1 is present or to part b) ii) if the TDOA estimate from the autocorrelation at receiver 2 is present.

**Case 5:** No TDOA estimate from the autocorrelation at both receivers is present

The only thing to do here is use the relationship

$$D_{d1d2} + D_{s1s2} = D_{s1d2} + D_{d1s2}$$

to classify the TDOA estimates of each source. If one or more of the four TDOA estimates from the cross-correlation for one source is missing then classification for that source is not possible.

After the TDOA estimates are classified to the corresponding source, they will be compared to each other and registered to the path pair signals that produce each
TDOA estimate based on the following ascending order in time

\[ D_{s1d2} > D_{s1s2} > D_{d1d2} > D_{d1s2}, \]

if \( \{D_{s1s2}, D_{d1d2}\} \) have positive sign and

\[ D_{s1d2} > D_{d1d2} > D_{s1s2} > D_{d1s2} \]

if \( \{D_{s1s2}, D_{d1d2}\} \) have negative sign.

The software code for implementing the registration algorithm in MATLAB is included in Appendix A.3.

### 6.4 Comparing Simulation and Actual Results

#### 6.4.1 Source of Errors

The exact results from the simulation are not possible because of the interpolations involved in the simulation process. The Doppler's effect causes time scale compression and expansion and yet the received signal must be a uniformly sampled sequence. The values must be interpolated at the new sample times. Furthermore, the sample times will not correspond with known transmitted points; the time when a sampled point had been transmitted must be interpolated as well.

The actual differential time delay is approximated with the difference in the arrival times of the first point in each transmission interval. This is known to be imperfect because extractions are unlikely to be consistent near the first point in each interval. The relative time companding is approximated with the average difference of the differential time delay during each transmission interval. All values obtained in the implementation of the algorithm are not likely to coincide with the sampled data point or with the step size set in the search for the correlation peaks and have to be interpolated.

Nevertheless, all interpolation methods used are relatively accurate and sufficient for this work. The comparison is done for the purpose of proving the concept; thus, the results from implementing the algorithm will be compared relatively to the actual values.
There are a number of sources of error for the results.

i) The amount of time shifting is not a whole number and has to be rounded to the nearest integer. This quantizing error is independent from the correlation function and causes a shift in the peaks in the individual correlation functions.

ii) The step size of differential time delay and differential time delay rate in the search can also affect the results. Smaller step size will reduce the result error.

iii) The noise in the received signal due to all the path signals and filter noise reduces the accuracy of the result.

vi) The acceleration term is small and assumed to be zero for a short time interval due to the assumption of linear varying time delay. This acceleration term is different for each pair of path signals depending on the source location and velocity. Over a long period of time this acceleration term accumulates and becomes more significant for one pair of path signals than the others. This results in different errors for the TDOA estimates.

v) The correlation peaks of the source may be affected due to the presence of the peaks from the interference. This problem tends to get worse if the peaks are positioned closely on the PAS.

6.4.2 Comparison of Actual Value and Simulated Results

There are two general cases that are considered in this comparison. The locations of the sensors $d_1$, $d_2$, $s_1$ and $s_2$ are the same for both cases which are given, respectively, by

\[ x_{d_1} = [x_{d_1}, y_{d_1}, z_{d_1}] = [500, 0, 200] \]
\[ x_{d_2} = [x_{d_2}, y_{d_2}, z_{d_2}] = [-500, 0, 200] \]
\( \mathbf{x}_{s1} = [x_{s1}, y_{s1}, z_{s1}] \)
\( = [500, 0, -200] \)
\( \mathbf{x}_{s2} = [x_{s2}, y_{s2}, z_{s2}] \)
\( = [-500, 0, -200] \)

In the first case the correlation peaks on the PAS are positioned relatively far apart and the resulting peaks can be obtained without any severe effect from each other. The source initial location and velocity used in the first trial are

\( \mathbf{x}_c = [x_c, y_c, z_c] \)
\( = [-1200, 200, 300] \)
\( \mathbf{v}_c = [V_{xc}, V_{yc}, V_{zc}] \)
\( = [15, 0, 0] \)

where \( \mathbf{v}_c \) the source velocity vector is specified in the program as speed \( V_c = 15m/s \) and direction vector \([1, 0, 0]\). The interferer initial location and velocity used in this trial are

\( \mathbf{x}_I = [x_I, y_I, z_I] \)
\( = [-800, 800, 300] \)
\( \mathbf{v}_I = [V_{xI}, V_{yI}, V_{zI}] \)
\( = [10, 0, 0] \)

where \( \mathbf{v}_I \) is the interferer velocity vector specified in the program as speed \( V_i = 10m/s \) and direction vector \([1, 0, 0]\).

Table 6.1 shows a comparison between the actual values and simulated results of the path pair differential time delays implementing both algorithms using MATLAB for both the source and the interferer.

Simulated results from implementing both algorithms agree with actual values with some acceptable errors.

In the second case, the same source initial location and velocity are used but
Table 6.1: Comparison between actual values and simulated results for the first trial

<table>
<thead>
<tr>
<th>Delay</th>
<th>$D_{d1d2}$ (in ms)</th>
<th>$D_{d1s2}$ (in ms)</th>
<th>$D_{s1d2}$ (in ms)</th>
<th>$D_{s1s2}$ (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Value</td>
<td>652.528</td>
<td>551.140</td>
<td>698.908</td>
<td>597.404</td>
</tr>
<tr>
<td>Resulting Value from implementing Algorithm II</td>
<td>652.490</td>
<td>551.379</td>
<td>698.998</td>
<td>597.682</td>
</tr>
<tr>
<td>Error</td>
<td>+0.038</td>
<td>-0.239</td>
<td>-0.090</td>
<td>-0.278</td>
</tr>
<tr>
<td>Resulting Value from implementing Algorithm I</td>
<td>652.671</td>
<td>551.017</td>
<td>698.963</td>
<td>597.669</td>
</tr>
<tr>
<td>Error</td>
<td>-0.143</td>
<td>+0.123</td>
<td>-0.055</td>
<td>-0.265</td>
</tr>
</tbody>
</table>

For the Interferer

<table>
<thead>
<tr>
<th>Delay</th>
<th>$D_{d1d2}$ (in ms)</th>
<th>$D_{d1s2}$ (in ms)</th>
<th>$D_{s1d2}$ (in ms)</th>
<th>$D_{s1s2}$ (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Value</td>
<td>443.886</td>
<td>357.349</td>
<td>495.023</td>
<td>408.467</td>
</tr>
<tr>
<td>Resulting Value from implementing Algorithm I</td>
<td>444.174</td>
<td>357.203</td>
<td>495.201</td>
<td>408.393</td>
</tr>
<tr>
<td>Error</td>
<td>-0.288</td>
<td>+0.146</td>
<td>-0.178</td>
<td>-0.074</td>
</tr>
<tr>
<td>Resulting Value from implementing Algorithm II</td>
<td>444.072</td>
<td>357.466</td>
<td>495.127</td>
<td>408.623</td>
</tr>
<tr>
<td>Error</td>
<td>-0.186</td>
<td>-0.117</td>
<td>-0.104</td>
<td>-0.156</td>
</tr>
</tbody>
</table>
the initial location of the interferer is changed so that the correlation peaks of the source and the interferer on the PAS are positioned relatively close to each other in pairs. The source initial location and velocity used in the second trial are

\[ \mathbf{x}_c = [x_c, y_c, z_c] \]
\[ = [-1200, 200, 300] \]
\[ \mathbf{V}_c = [V_{xc}, V_{yc}, V_{zc}] \]
\[ = [15, 0, 0] \]

where \( \mathbf{V}_c \) is the source velocity vector specified in the program as speed \( V_c = 15\text{m/s} \) and direction vector \([1, 0, 0]\). The interferer initial location and velocity used in this trial are

\[ \mathbf{x}_I = [x_I, y_I, z_I] \]
\[ = [-1200, -180, 300] \]
\[ \mathbf{V}_I = [V_{xI}, V_{yI}, V_{zI}] \]
\[ = [10, 0, 0] \]

where \( \mathbf{V}_I \) is the interferer velocity vector specified in the program as speed \( V_I = 10\text{m/s} \) and direction vector \([1, 0, 0]\).

Table 6.2 shows a comparison of the actual values and simulated results of the path pair differential time delay for the source and the interferer. The interference effect is expected to be more pronounced in this case. Simulated results agree with actual values with some errors which are greater than the errors in the first trial.

Most of the sources of error are independent and sometime they might counteract each other; thus, the error in the results might be reduced or increased depending on how the sources of error interact with each other. This can be seen from the results in Table 6.2. The errors for the source in some path pair differential time delays reduce while the error of others increase.

Although the correlation peaks are positioned pairwise close together on the differential time delay axis of the PAS, the velocity difference separates them on the
Table 6.2: Comparison between actual values and simulated results for the second trial

<table>
<thead>
<tr>
<th>Delay</th>
<th>$D_{d1d2}$ (in ms)</th>
<th>$D_{d1s2}$ (in ms)</th>
<th>$D_{s1d2}$ (in ms)</th>
<th>$D_{s1s2}$ (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Value</td>
<td>652.528</td>
<td>551.140</td>
<td>698.908</td>
<td>597.404</td>
</tr>
<tr>
<td>Resulting Value from implementing Algorithm II</td>
<td>652.469</td>
<td>551.253</td>
<td>698.728</td>
<td>597.435</td>
</tr>
<tr>
<td>Error</td>
<td>+0.059</td>
<td>-0.113</td>
<td>+0.180</td>
<td>-0.031</td>
</tr>
<tr>
<td>Resulting Value from implementing Algorithm I</td>
<td>652.721</td>
<td>551.319</td>
<td>698.834</td>
<td>597.582</td>
</tr>
<tr>
<td>Error</td>
<td>-0.193</td>
<td>-0.179</td>
<td>+0.074</td>
<td>-0.178</td>
</tr>
</tbody>
</table>

For the Interferer

<table>
<thead>
<tr>
<th>Delay</th>
<th>$D_{d1d2}$ (in ms)</th>
<th>$D_{d1s2}$ (in ms)</th>
<th>$D_{s1d2}$ (in ms)</th>
<th>$D_{s1s2}$ (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Value</td>
<td>654.800</td>
<td>553.637</td>
<td>701.026</td>
<td>599.786</td>
</tr>
<tr>
<td>Resulting Value from implementing Algorithm II</td>
<td>655.054</td>
<td>553.951</td>
<td>701.280</td>
<td>599.810</td>
</tr>
<tr>
<td>Error</td>
<td>-0.254</td>
<td>-0.314</td>
<td>-0.254</td>
<td>-0.024</td>
</tr>
<tr>
<td>Resulting Value from implementing Algorithm I</td>
<td>654.544</td>
<td>553.870</td>
<td>701.158</td>
<td>600.045</td>
</tr>
<tr>
<td>Error</td>
<td>+0.059</td>
<td>-0.233</td>
<td>-0.132</td>
<td>-0.259</td>
</tr>
</tbody>
</table>
differential time delay rate axis; therefore, it is still possible to detect these peaks separately. The worst case is where the source and the interferer have the same initial location and velocity then they can only be located as a single source.
7. Summary and Conclusions

A critical aspect of the underwater acoustic source localization problem is to correctly register the time delay path pair estimates. The main focus in this thesis is to develop robust algorithms for the time delay path pair estimation and association problem of multiple moving sources in a multipath environment using pairs of sensors. Initially however, the work will focus on the problem of two different sources moving on straight line paths past two receivers.

Received signals at the two sensors from remote acoustic sources are simulated and used throughout in this work. A model of the ocean transmission for the received signals is presented with reasonable assumptions for simplification. The model includes a pre-set sensors location configuration in the ocean as a reference for the source initial location. Each source is moving on a straight line path with a constant velocity which can be set in each simulation trial. Simulated data generated in one-second interval introduces some complexity for the benefit of keeping the size of the program small. Noise added to the received signal is generated at a constant power level using the same filter as used for the signal. The simulation program also keeps track of the time delay of each data second for the purpose of comparison.

An expression for the expected value of time average cross-correlation is developed. Reasonable simplifying assumptions are used to produce a more manageable expression without significant degradation. The resulting equation is a function of the source and interference signal power spectra. Simulated results by implementing the equation in MATLAB indicates that the position of the main lobe in the TDOA peak is affected by the side lobes of TDOA peaks of the interference. This effect results in a time shift of the TDOA peak and the amount of time shifting is dependent on how
close the positions of interference peaks are.

Algorithm I, which is based on the Select-Correlate-Sum technique, is suggested as an alternative algorithm. Algorithm I is similar to the proposed algorithm, Algorithm II, where the main difference is in the sequence of execution. The proposed TDOA estimation algorithm is developed based on the Deskewed Short-Time Correlator technique (DSTC). The DSTC is preferred over other techniques for its advantage of significant reduction in computational time (about 100 times in the simulation). Simulated results shows Algorithm II provides similar accuracy compared to that provided by Algorithm I.

The TDOA estimates registration algorithm is developed by using the relationships of the TDOA estimates obtained from the autocorrelation and cross-correlation. Two location sources (symmetric over the x-axis) will have the same TDOA estimates, thus only one half of the total search space needs to be examined. Simple geometric theorems are used to show that the TDOA estimates from the detected source in each quadrant of the examining half of the space follow an ascending order in time. The relationships of resulting TDOA estimates from the autocorrelation and cross-correlation are used to classify the TDOAs to the corresponding source and then register them to the pairs of path signals that produce each TDOA based on the ascending order in time. Several pairs of sensors are needed to locate the source to a single position.

From the work done in this report it is concluded that

1. The expression obtained for the expected value of the path pair TDOA estimates shows the TDOA estimates are affected by the presence of the interference. The degree of effect is increased when the interference and the contact are located closely. The side lobes of the interferer correlogram could be large enough to affect the main lobe of the contact correlogram. This leads to displacing of the peak of the main lobe for the contact correlogram which in turn affects the accuracy of the results. Simulated results show that there is an in-
crease in errors for the TDOA estimates obtained from the contact when the interference is located closely to the source.

2. The Deskewed Short-Time Correlator technique can be used in the proposed TDOA estimation algorithm, Algorithm II, providing similar accuracy and significant reduction in computational time (about one hundred times less correlation in the simulation).

3. The short time correlation algorithm is capable of isolating multiple sources by reducing the effect of interference which improves the resulting TDOA estimates.

4. The Interval-in-Interval correlation technique can be used in addition with the short time correlation technique to reduce the effect of interference further.

5. Simulated results using MATLAB indicate that the developed algorithms are capable of performing the TDOA estimation and registration for two moving sources in a multipath environment.
References


A. Software Code

A.1 Signal Simulation

This simulation is an implementation of the transmission system model in Chapter 2 using MATLAB programming. The programs used for this simulation are modified versions of K. Jeffrey's programs to include the multipath case. The received signal can be generated in a single path or a multipath environment using a pair of receivers for a single source with or without interference.

A.1.1 Program Structure

The structure of the programs can be shown by a tree diagram in Figure A.1. Starting at the top of the tree diagram is the main program which will be executed to start the signal simulation. Below the main programs are the subprograms and functions that will be called upon execution.

A.1.2 Usage

All programs specified in the list for the signal simulation must be placed in the same directory for the simulation. For clarity all file names and variables will be italic. The received signals are generated by executing the file named mainprog.m. Before executing mainprog.m there are a few parameters that have to be specified in this file.

- Number of sensors: Enter the number of sensors by setting a value for sensors (maximum 2)
- Number of sources: Enter the number of sources by setting a value for nrofsource
Figure A.1: Tree diagram of the programs in implementing the received signal simulation
(1 for a single source without interference and 2 for a single source with interference)

- Number of paths: Enter the number of paths by setting a value for \textit{nrofpath} (1 for direct path only and 2 for both direct and surface paths)

- Source speed: Source speed is set from the main program by typing the number in meters per second for \textit{a.speed} indicating the speed of the source of interest or \textit{b.speed} indicating the speed of the interference source. The speed is limited to a maximum of 20m/s.

- Length of the received signal data sequence: Length of the received signal data sequence is specified by entering the number of data seconds \textit{nrofsecond}.

Other parameters can also be set in the subprograms and functions

- Source initial location and motion direction: The source initial location and motion direction can be set by change \textit{a.track} number for a single source without interference and both \textit{a.track} and \textit{b.track} for a single source with interference in the main program \textit{mainprog.m}. Each track number corresponds to a choice of initial location and motion direction preset in the two files \textit{track5.f.m} and \textit{velo5a.f.m}. The number of choices for initial location and motion direction is limited from 1 to 15. The initial location and motion direction entered in these two files for each source have to be the same.

- Sensors location: Default values for the sensors location are as following:
  
  Sensor Nr. 1: \(sens1 = [500, 0, 200]\),
  
  Sensor Nr. 2: \(sens2 = [-500, 0, 200]\)

  The sensor locations can be changed by changing the number for \textit{sens1} and \textit{sens2} for sensors 1 and 2, respectively at the beginning of the file:

  For multi-path environment received signals:
  
  2 sensors: \textit{gener6s2.f.m}

  For single-path environment received signals:
  
  2 sensors: \textit{gener62.f.m}

- Sound speed: The default value for the sound speed is 1500m/s. The sound speed is set in two files \textit{channel.f.m} and \textit{velo5a.f.m} by typing the number for the variable
sound_speed_g.

- Bounce loss for surface path: The bounce loss for surface path is initially set to -1. It can be set by changing the variable bounce_loss_g in the file channel_f.m with the value desired.

- Sampling frequency: The sampling rate for the received signal is set to be 8000 Hz but it can changed by typing the number for the variable sampling_freq_g in the following files for different cases appropriately
  
  For multipath 2 sensors: resamp6a_f.m, sim6s2.m and gener6s2_f.m
  For single path 2 sensors: resamp6a_f.m, sim62.m and gener62_f.m
  In addition the variable samp_freq_4x which is four times the sampling frequency should also be changed in the file source_2.m.

- Seed of random number generator: There is a preset value for the seed of the random generator but it can be changed by typing the following on the MATLAB command window:

  randseed = seed value  
  save randseed randseed

After all information are specified, the signal simulation programs can be executed by typing:

  mainprog (Enter)

on the MATLAB command window.

A.1.3 Output Format

The simulated received signals are stored in recsig1.mat and recsig2.mat for the two receivers 1 and 2, respectively with the same variable recsig. Noise data generated will be stored in the files noise1.mat and noise2.mat for the two receivers 1 and 2, respectively with the same variable noise. The received signal with noise can be generated simply by adding the noise to the signal. The results are stored in the files recsignoise1.mat and recsignoise2.mat for the two receivers 1 and 2 respectively and again with the same variable recsig.
The nomenclature for the data seconds are:

- A1D.01 source A, sensor 1, direct path, received interval 1 – 2 sec
- A1D.02 source A, sensor 1, direct path, received interval 2 – 3 sec
- A1S.01 source A, sensor 1, surface path, received interval 1 – 2 sec
- A2D.01 source A, sensor 2, direct path, received interval 1 – 2 sec
- A2.01 source A, sensor 2, combined direct and surface path received interval 1-2 sec
- B1D.01 source B, sensor 1, direct path, received interval 1 – 2 sec
- B1D.02 source B, sensor 1, direct path, received interval 2 – 3 sec
- B1S.01 source B, sensor 1, surface path, received interval 1 – 2 sec
- B2D.01 source B, sensor 2, direct path, received interval 1 – 2 sec
- B2.01 source B, sensor 2, combined direct and surface path received interval 1-2 sec
- Recl.01 sensor1, combined sources and paths, received 1 – 2 sec

Each file has a structure:

\[
\text{size(Subsamples)} = 8000 \text{ rows and 2 columns [value, time]}
\]

for combined sources and paths data second the structure is:

\[
\text{size(Subsamples)} = 8000 \text{ rows and 1 columns [value]}
\]

Generated data seconds starts from second 2 to make it easier when combining received signal from two different sources. The received signals with and without noise added are stored with the same structure:

\[
\text{size(recessig) = 1 row of data}
\]

For each source there will be 3 files saved to keep track of the path delays, source location and velocity. They are Adelay, B Trace and B VELO. All the time units are given as 1/sampling frequency in this case 1/8000 second. So a delay of 4000 means 0.5 second. The path pair differential time delay and the relative time companding have to be calculated manually with available data from Adelay.

A.1.4 Program Code

The following programs are written for UNIX environment. Documentation is included in the file with a % sign at the beginning of the first line.
1. File name: align_f.m
% ALIGN_F: align the start of buffer to an integral second
% Usage: [BUFFER, bufend, secnum] = align_f(BUFFER, bufend);
% where the first element of buffer returned will be the first sample
% after an integral second and secnum indicates which second it was.
% This is done so that it is easy to add multiple sources. Note that
% time 0 == isec will subsequently be second number 0.

function [BUFFER, bufend, secnum] = align_f(BUFFER, bufend)
first = buffer(1,1);
secnum = ceil(first);
after = find(buffer(:,1) >= secnum);
start = after(1);
count = bufend - start + 1;
BUFFER(start:count,:) = BUFFER(start:bufend,:);
bufend = count;
fprintf('Aligned received data to Second Number %2.0f\n',secnum);
end of align_f.m

2. File name: bufload_f.m
% BUFLOAD_F: append new simulated data to end of (ring) BUFFER
% Usage: [BUFFER, bufend] = bufload_f(BUFFER, bufend, NEWVEC);
% where BUFFER = [Arrtime Arrval Txtime] and so does NEWVEC

function [BUFFER, bufend] = bufload_f(BUFFER, bufend, NEWVEC)
newsize = length(NEWVEC);
BUFFER(bufend+1:bufend+newsize,0 = NEWVEC;
bufend = bufend + newsize;
end of bufload_f.m

3. File name: channel_f.m
% CHANNEL_F: determine what the path will do to any source signal
% Usage: CHANNEL = channel_f(Sensor_pos, TRACK, path_type)
% where CHANNEL = [Delay Gain], Sensor_pos is the coordinates
% (meters) of one sensor, path_type is a switch for either (1) direct
% or (2) surface bounce path

function CHANNEL = channel_f(Sensor_pos, TRACK, path_type)
direct_path_g = 1; % direct and surface path
sound_speed_g = 1500; % sound speed underwater
bounce_loss_g = -1; % Surface bounce loss
x_comp = 1; % index for x component
y_comp = 2; % index for y component
z_comp = 3; % index for z component
% Determine the RELATIVE path between source and sensor
REL_TRACK(:,x_comp) = TRACK(:,x_comp) - Sensor_pos(x_comp);
REL_TRACK(:,y_comp) = TRACK(:,y_comp) - Sensor_pos(y_comp);
if path_type == direct_path_g,
REL_TRACK(:,z_comp) = TRACK(:,z_comp) - Sensor_pos(z_comp);
elseif path_type == surface_path_g,
REL_TRACK(:,z_comp) = TRACK(:,z_comp) - (-1 * Sensor_pos(z_comp));
else
disp('Unknown path_type in Channel_f')
end
% Find the Euclidian distances between source and sensor
Euclid = sqrt(sum(REL_TRACK' * REL_TRACK));
% use the distance to get the delay introduced by this channel
Delay = Euclid ./ sound_speed_g;
% Determine spreading loss for this channel
Gain = 1 ./ (Euclid); % factor by signal reduced due to spreading loss
if this is a surface-bounce path then add bounce loss
if path_type == surface_path_g,
Gain = Gain .* bounce_loss_g;  % bounce_loss_g further fraction
end
CHANNEL = [Delay Gain];
% end of channel_f.m

4. File name: comb_f.m
% COMB_F: add signal from two sources. Data is read as 1 second files
% add only time-matching files. Save again as files without Txtime data

function comb_f(src1, src2, sensors, endsec)
for second = 0:endsec,
    for sensor = 1:sensors,
        if second < 10,
            str1 = eval(['A',int2str(sensor),'0',int2str(second),'.mat']);
            str2 = eval(['B',int2str(sensor),'0',int2str(second),'.mat']);
        else
            str1 = eval(['A',int2str(sensor),int2str(second),'.mat']);
            str2 = eval(['B',int2str(sensor),int2str(second),'.mat']);
        end
        if (exist(str1)==2) && (exist(str2)==2),
            eval(['load ',str1])
            Asig = Subsamples(:,1);  % omit Txtime data
            eval(['load ',str2])
            Bsig = Subsamples(:,1);
            fprintf('Combining sensor %1.0f in Second Number %2.0f
',sensor,second);
            Subsamples = Asig + Bsig;
        end
    end
end
% end of comb_f.m

5. File name: filter_f.m
% FILTER_F: designs a low-pass filter
% Usage: [FILTER,filter_energy] = filter_f(fs) where fs is sampling
% frequency.

function [FILTER,filter_energy] = filter_f(fs)
    nyq = fs/2;
    passband = 400/nyq;  % passband freq as fraction of Nyquist in Hertz
    stopband = 500/nyq;  % stopband freq as fraction of Nyquist in Hertz
    passdrop = 1;  % tolerance in passband. <= -1 dB
    stopdrop = 15;  % tolerance in stopband. >= -15 dB
    % Design a lowpass filter. Where n is the filter order required and
    % wn is the actual cutoff frequency
    [n,wn] = buttord(passband, stopband, passdrop, stopdrop);
    [B,A] = butter(n,wn);
    % Determine how much power the filter will pass by integrating its
    % magnitude-squared response. Note that h is really a periodogram for
    % a filter with a bandwidth of 1 Hz.
    freq_slices = 8192;
    [H,W] = freqz(B,A,freq_slices);
    filter_energy = sum(abs(H).^2 .* 1/freq_slices);
    FILTER = [A;B];
% end of filter_f.m

6. File name: gener62_f.m
% GENER62_F: generate signal and path and propagate signal along
% path (single path)
function [PATH_1D, PATH_2D, Tail, VELO] = ...
    gener62_f(iter, src, track_choice, speed, Tail, VELO)
fprintf('Generating and propagating more data
');
sampling_freq_g = 8000; % sampling frequency
direct_path_g = 1; % direct and surface path
surface_path_g = 2;
Sens1 = [500 0 200]; % Sensors location
Sens2 = [-500 0 200];
interval_g = 1; % length of each signal time interval generated in second
testsig_g = 0; % if testsig_g = 1, generate a sine wave
% and not random signal
% generate one second of new time base
Windowlx = [1:sampling_freq_g]'; % samples for 1 second
Time = Windowlx + (iter-1)*round(interval_g*sampling_freq_g); % in samples
TRACK = track5_f(Time, speed, track_choice, sampling_freq_g);
% Save some special location information as a trace of the motion
if (src == 1),
    load A_Trace
    A_Trace = [A_Trace; TRACK(1,:)];
    save A_Trace A_Trace
else
    load B_Trace
    B_Trace = [B_Trace; TRACK(1,:)];
    save B_Trace B_Trace
end
% Compute "VELOCITY" information (i.e., delay changing rate)
VELO_1D = velo5a_f(Sens1, TRACK, speed, track_choice, direct_path_g);
VELO_2D = velo5a_f(Sens2, TRACK, speed, track_choice, direct_path_g);
VELO = [VELO; [VELO_1D VELO_2D]];
function [PATH_1D, PATH_2D, Tail, VELO] = ...
    gener5_f(iter, src, track_choice, speed, Tail, VELO)
fprintf('Generating and propagating more data
');
sampling_freq_g = 8000; % sampling frequency
direct_path_g = 1; % direct and surface path
surface_path_g = 2;
Sens1 = [500 0 200]; % Sensors location
Sens2 = [-500 0 200];
interval_g = 1; % length of each signal time interval generated in second
testsig_g = 0; % if testsig_g = 1, generate a sine wave
% and not random signal
% generate one second of new time base
Windowlx = [1:sampling_freq_g]'; % samples for 1 second
Time = Windowlx + (iter-1)*round(interval_g*sampling_freq_g); % in samples
TRACK = track5_f(Time, speed, track_choice, sampling_freq_g);
% Save some special location information as a trace of the motion
if (src == 1),
    load A_Trace
    A_Trace = [A_Trace; TRACK(1,:)];
    save A_Trace A_Trace
else
    load B_Trace
    B_Trace = [B_Trace; TRACK(1,:)];
    save B_Trace B_Trace
end
% Compute "VELOCITY" information (i.e., delay changing rate)
VELO_1D = velo5a_f(Sens1, TRACK, speed, track_choice, direct_path_g);
VELO_2D = velo5a_f(Sens2, TRACK, speed, track_choice, direct_path_g);
VELO = [VELO; [VELO_1D VELO_2D]];
function [PATH_1D, PATH_2D, Tail, VELO] = ...
    gener5_f(iter, src, track_choice, speed, Tail, VELO)
fprintf('Generating and propagating more data
');
sampling_freq_g = 8000; % sampling frequency
direct_path_g = 1; % direct and surface path
surface_path_g = 2;
Sens1 = [500 0 200]; % Sensors location
Sens2 = [-500 0 200];
interval_g = 1; % length of each signal time interval generated in second
testsig_g = 0; % if testsig_g = 1, generate a sine wave
% and not random signal
% generate one second of new time base
Windowlx = [1:sampling_freq_g]'; % samples for 1 second
% Function to compute the required FILTER
function [FILTER, filter_energy] = filter_f(sampling_freq_g);
    [FILTER, filter_energy] = filter_f(sampling_freq_g);
end
[Source, Tail] = source_2(Time, FILTER, filter_energy, testsig_g, Tail);
clear CHANNEL_1D
PATH_1D = path_f(Source, CHANNEL_1D, Time, sampling_freq_g);
clear CHANNEL_1D
PATH_2D = path_f(Source, CHANNEL_2D, Time, sampling_freq_g);
end of gener62_f.m

7. File name: gener6s2_f.m
% GENER6S2_F: generate signal and path and propagate signal along
% path (multipath, two sensors)
function [PATH_1D, PATH_2D, PATH_1S, PATH_2S, Tail, VELO] = ...
    gener5_f(iter, src, track_choice, speed, Tail, VELO)
fprintf('Generating and propagating more data \n');
sampling_freq_g = 8000; % sampling frequency
direct_path_g = 1; % direct and surface path
surface_path_g = 2;
Sens1 = [500 0 200]; % Sensors location
Sens2 = [-500 0 200];
interval_g = 1; % length of each signal time interval generated in second
testsig_g = 0; % if testsig_g = 1, generate a sine wave
% and not random signal
% generate one second of new time base
Windowlx = [1:sampling_freq_g]'; % samples for 1 second
% Compute CHANNEL which introduces delay and loss
CHANNEL_1D = channel_f(Sens1, TRACK, direct_path_g);
CHANNEL_2D = channel_f(Sens2, TRACK, direct_path_g);
clear TRACK
% Make the required FILTER, then use it to create the source signal
if exist('sea2filt.mat') == 2,
    load sea2filt
else
    [FILTER, filter_energy] = filter_f(sampling_freq_g);
    save sea2filt FILTER filter_energy
end
[Source, Tail] = source_2(Time, FILTER, filter_energy, testsig_g, Tail);
clear CHANNEL_1D
PATH_1D = path_f(Source, CHANNEL_1D, Time, sampling_freq_g);
clear CHANNEL_1D
PATH_2D = path_f(Source, CHANNEL_2D, Time, sampling_freq_g);
end of gener6s2_f.m
Time = Time + (iter - 1) * round(interval_g * sampling_freq_g); \% in samples
 TRACK = track5_f(Time, speed, track_choice, sampling_freq_g);
\% Save some spatial location information as a trace of the motion
if (src == 1),
 load A_Trace
 A_Trace = [A_Trace; TRACK(1,:)];
 save A_Trace A_Trace
else
 load B_Trace
 B_Trace = [B_Trace; TRACK(1,:)];
 save B_Trace B_Trace
end
\% Compute "VELOCITY" information, i.e. delay changing rate
VELO_1D = velo5a_f(Sens1, TRACK, speed, track_choice, direct_path_g);
VELO_2D = velo5a_f(Sens2, TRACK, speed, track_choice, direct_path_g);
VELO_1S = velo5a_f(Sens1, TRACK, speed, track_choice, surface_path_g);
VELO_2S = velo5a_f(Sens2, TRACK, speed, track_choice, surface_path_g);
VELO = [VELO ; [VELO_1D VELO_2D ]];
\% Compute the CHANNEL which introduces delay and loss
CHANNEL_1D = channel_f(Sens1, TRACK, direct_path_g);
CHANNEL_2D = channel_f(Sens2, TRACK, direct_path_g);
CHANNEL_1S = channel_f(Sens1, TRACK, surface_path_g);
CHANNEL_2S = channel_f(Sens2, TRACK, surface_path_g);
clear TRACK
\% Make the required FILTER, then use it to create the source signal
if exist('sea2filt.mat')==2,
 load sea2filt
else
 [FILTER, filter_energy] = filter_f(sampling_freq_g);
 save sea2filt FILTER filter_energy
end
[Source, Tail] = source_2(Time, FILTER, filter_energy, testsig_g, Tail);
clear FILTER filter_energy
\% Push the source signal through the CHANNEL to get the received sequence
PATH_1D = path_f(Source, CHANNEL_1D, Time, sampling_freq_g);
clear CHANNEL_1D
PATH_1S = path_f(Source, CHANNEL_1S, Time, sampling_freq_g);
clear CHANNEL_1S
PATH_2D = path_f(Source, CHANNEL_2D, Time, sampling_freq_g);
clear CHANNEL_2D
PATH_2S = path_f(Source, CHANNEL_2S, Time, sampling_freq_g);
end of gener62s_f.m

8. File name: mainprog.m
\% MAINPROG: Main program for the received signals simulation

A = 1; \% Index for the sources
B = 2;
sensors = 2; \% number of sensors
nrofsource = 2; \% number of sources (maximum 2)
nrofpath = 2; \% 1 for direct paths only, 2 for both direct and \% surface paths
nrofsecond = 2; \% number of simulated data seconds
endsec = nrofsecond * 2;
\% source A
a_track = 14;
a_speed = 15;
\% source B
b_track = 15;
b_speed = 5;
elseif nrofsource == 2 & sensors == 2 & nrofpath == 2
signalgener22
elseif nrofsource == 2 & sensors == 2 & nrofpath == 1
signalgener22d
elseif nrofsource == 1 & sensors == 2 & nrofpath == 2
signalgener12
elseif nrofsource == 1 & sensors == 2 & nrofpath == 1

signalgenerator2d
else
fprintf('exceeding parameters\n');
end
% end of mainprog.m

9. File name: multi_f.m
% Usage: multi_f(src,double, endsec);
% where src is 1 for A endsec says which is the last seg to combine.
% MULTI_F: add signal from two paths. Data is read as 1 second files.
% add only time-matching files. Void Ttime data.

function multi_f(src, sensors, endsec)
for second=0:endsec,
    for sensor=1:sensors,
        numstr = ['0' int2str(second)];
        numstr = numstr(length(numstr)-1:length(numstr));
        if src==1,
            strD = eval(['A' int2str(sensor) 'D_' numstr '.mat']);
            strS = eval(['A' int2str(sensor) 'S_' numstr '.mat']);
        else
            strD = eval(['B' int2str(sensor) 'D_' numstr '.mat']);
            strS = eval(['B' int2str(sensor) 'S_' numstr '.mat']);
        end
        if ((exist(strD)==2) & (exist(strS)==2)),
            eval(['load ',strD])
            Directsig = Subsamples;
            eval(['load ',strS])
            Surfacesig = Subsamples;
            fprintf('Combining sensor %d Paths in Second Number %d
', sensor, second);
            Subsamples = Directsig + Surfacesig;
            if src==1,
                str = eval(['A' int2str(sensor) 'numstr '.mat']);
            else
                str = eval(['B' int2str(sensor) 'numstr '.mat']);
            end
            eval(['save ',str,' Subsamples'])
        end
    end
end
% end of multi_f.m

10. File name: noisecomb_f.m
% SIGCOMB_F: adds noise to signal sequence

function noisecomb_f(sensors, endsec)
load A1_03;
1 = length(Subsamples);
for sensor = 1:sensors,
    for second = 2:endsec-1,
        if second<10,
            load(eval(['A',int2str(sensor),'_0',int2str(second),' ']))
            noise((1+(second-2)):(1+(second - 2)))=Subsamples(:,1);
        else
            load(eval(['A',int2str(sensor),'_1',int2str(second),' ']))
            noise((1+(second-2)):(1+(second - 2)))=Subsamples(:,1);
        end
        str=eval(['noise ',int2str(sensor),' ']);
        eval(['save ',str,' noise '])
    end
end
% end of noisecomb_f.m

11. File name: path_f.m
function PATH = path_f(Source, CHANNEL, Timebase, freq)
  delay_comp = 1; % index for delay vector in CHANNEL matrix
  gain_comp = 2; % index for gain vector in CHANNEL matrix
  Txtime = Timebase ./ freq;
  Arrtime = (Timebase ./ freq) + CHANNEL(:, delay_comp);
  Arrsig = Source .* CHANNEL(:, gain_comp);
  PATH = [Arrtime Arrsig Txtime];
end

function [RESAMP, NEWBUF, newend] = resamp6a_f(PATHBUF, bufend, segnum, sens)
  sampling_freq_g = 8000; % sampling frequency
  arrtim = 1; % Names for columns of PATHBUF
  range = 80; % points in search space required beyond 1 full segment
  Window1x = [0:sampling_freq_g-1] / sampling_freq_g;
  NEWtim = Window1x + segnum;
  Newtim = Window1x + segnum;
  if (PATHBUF(bufend, arrtim) < (NEWtim(length(NEWtim)) + range / sampling_freq_g)),
    error('subsamp4_f: ran out of data in buffer');
  end
  % Check that the time segments are what we think they are:
  if (PATHBUF(1, arrtim) - NEWtim(1) > 2 / sampling_freq_g),
    error('subsample aligned problem');
  end
  % For each (re)sample time in Newtim, find the first next (arrive)
  % sample time in PATHBUF(:, 1), and use PATHBUF points on either side to
  % compute the sinx/x interpolation.
  % Next is a vector of indexes into PATHBUF, which meet the criterion.
  segstart = sumpts + 1;
  current = find(PATHBUF(1:bufend, 1) < (segnum + 1));
  segend = current(length(current));
  t = 1 / sampling_freq_g;
  RESAMP = zeros(length(NEWtim), 2);
  RESAMP(:, 1) = [1:length(NEWtim)];
  range = 80:200; % limit search in PATHBUF(:, arrtim)
  % to +/- 30 pts for speed
  tick = 0; for each resample point in segment
  for tick = 1:length(Newtim),
    check = Newtim(tick); % find the time of this resample point
    test = tick + segstart; % find the index into PATHBUF of
    the next largest time
    Afind = find(PATHBUF(test + range, arrtim) >= check);
    Atest = test + range(Afind);
    % Sum sinx/x for sumpts on either side of located arrival time
    TDiff = check - PATHBUF(Atest(1)) - sumpts; Atest(1) + sumpts - 1, arrtim);
    Phi = sin(TDiff * pi / t);
    % sometimes the arrival and resample times are the same
    % => divide by zero fix these by substituting one,
    % since sin(x) as x->0 tends to one????
    divzerod = find(isnan(Phi));

end
if (length(divzerod)>0),
    Phi(divzerod) = ones(1,length(divzerod));
end
RESAMP(tick,arrval) =
    sum(PATHBUF(Atest(1)-sumpts:Atest(1)+sumpts-1,arrval).*Phi);
% Estimate tx time of resample using linear interp of actual
arrival times
    interp_frac = (PATHBUF(Atest(1),arrtim) - check) /
    (PATHBUF(Atest(1),arrtim) - PATHBUF(Atest(1)-1,arrtim));
    RESAMP(tick,txtim) = PATHBUF(Atest(1),txtim) - interp_frac *
    (PATHBUF(Atest(1),txtim) - PATHBUF(Atest(1)-1,txtim));
end
% Discard already resampled part of PATHBUF; grab remaining
% part into NEWBUF
    grab = [segend+l-sumpts:bufend]';
    NEWBUF(1:length(grab),:) = PATHBUF(grab,:);
newend = length(NEWBUF);
return
% end of resamp6a_f.m

13. File name: savesub6_f.m
% SAVESUB_F: write the subsampled second to file: [value Txtime]
% the filename indicates which second is contained.
% Usage: savesub_f(SUB, secnum); Where SUB = [Seq* Values Txtime]
function savesub_f(SUB, secnum, string)
    Subsamples = SUB(:,2:3);
    str = eval(['"saving ',string,int2str(secnum),'to file\n"']);
    fprintf(str);
    if secnum < 10,
        eval(['save ',string, '0',int2str(secnum), ' Subsamples']);
    else 
        eval(['save ',string, int2str(secnum), ' Subsamples']);
    end
% end of savesub6_f.m

14. File name: setBpower_f.m
% Usage: setBpower_f(sensors, endsec); set power ratio of source B
% and source A to -10 dB
function setBpower_f(sensors, endsec)
    for second=0:endsec,
        for sensor=1:sensors,
            numstr = ['0' int2str(second)];
            numstr = numstr(length(numstr)-1:length(numstr));
            strD = eval(['''B'' int2str(sensor) 'D_' numstr '.mat']);
            strS = eval(['''B'' int2str(sensor) 'S_' numstr '.mat']);
            if (exist(strD)==2)
                eval(['load ',strD])
                Subsamples(:,1) = Subsamples(:,1)./sqrt(10);
                eval(['save ','strD,' Subsamples']);
            end
            if (exist(strS)==2)
                eval(['load ','strS'])
                Subsamples(:,1) = Subsamples(:,1)./sqrt(10);
                eval(['save ','strS,' Subsamples']);
            end
        end
    end
% end of setBpower_f.m

15. File name: sigcomb2l_f.m
% SIGCOMB21_F: combines received data seconds at the sensors
% into one signal sequence (single path)
function sigcomb2l_f(endsec)
load A1.03;
1 = length(Subsamples);
for sensor = 1:2,
for second = 2:endsec-1,
if second<10,
load(eval(
['''A',int2str(sensor),'_0',int2str(second),''']))
recsig((1+1*(second-2)):1:1+(second-2)))=Subsamples(:,1);
else
load(eval(
['''A',int2str(sensor),'_',int2str(second),''']))
recsig((1+1*(second-2)):1:1+(second-2)))=Subsamples(:,1);
end
str=eval(
[''recsig','int2str(sensor),'''])
end
end
% end of sigcomb2l_f.m

16. File name: sigcomb2_f.m
% SIGCOMB2_F: combines received data seconds at the sensors
% into one signal sequence (multipath)

function sigcomb2_f(endsec)
load Rec1_03;
1 = length(Subsamples);
for sensor = 1:2,
for second = 2:endsec-1,
if second < 10,
load(eval(
[''Rec',int2str(sensor),'_0',int2str(second),''']))
recsig((1+1*(second-2)):1:1+(second-2)))=Subsamples;
else
load(eval(
[''Rec',int2str(sensor),'_',int2str(second),''']))
recsig((1+1*(second-2)):1:1+(second-2)))=Subsamples;
end
str=eval(
[''recsig','int2str(sensor),'''])
end
end
% end of sigcomb2_f.m

17. File name: signalgenerl2.m
% SIGNALGENER12: signal simulation for single moving source with
% two receivers in a multipath environment

fprintf('Starting simulation of source A\n');
sim6s2(A,a_track,a_speed,endsec);
fprintf('Combining paths for A\n');
multi_f(A,sensors,endsec);
fprintf('Starting to combine data seconds\n');
sigcomb2l_f(endsec);
fprintf('Starting simulation of noise with respect to source A\n');
clear a_speed;
a_speed=0;
sim6s2(A,a__track,a__speed,endsec);
fprintf('Starting to combine noise data seconds\n');
nosigcomb_f(sensors, endsec);
fprintf('Starting to add noise to received signals\n');
clear
load noisel
load recsigl
recsig=recsig+noise;
save recsignoise recsig
load noise2
load recsig2
recsig=recsig+noise;
101

save recsignoise2 recsig
clear
% end of signalgener12.m

18. File name: signalgener12d.m
% SIGNAGENER12D: Signal simulation for single moving source with
two receivers in single path environment

fprintf('Starting simulation of source A\n');
sim62(A,a_track,a_speed,endsec);
fprintf('Starting to combine data seconds\n');
sigcomb21_f(endsec);
fprintf('Starting simulation of noise with respect to source A\n');
clear a_speed;
a_speed=0;
sim62(A,a_track,a_speed,endsec);
fprintf('Starting to combine noise data seconds\n');
noisecomb_f(sensors, endsec);
fprintf('Starting to add noise to received signals\n');
clear
load noisel
load recsigi
recsig=recsig+noise;
save recsignoise1 recsig

clear
load noise2
load recsig2
recsig=recsig+noise;
save recsignoise2 recsig
clear
% end of signalgener12d.m

19. File name: signalgener22.m
% SIGNAGENER22: Signal simulation for two sources with two receivers
in a multipath environment

fprintf('Starting simulation of source A\n');
sim6s2(A,a_track,a_speed,endsec);
fprintf('Starting simulation of source B\n');
sim6s2(B,b_track,b_speed,endsec);
setBpower_f(sensors, endsec);
fprintf('Combining paths for A\n');
multi_f(A,sensors,endsec);
fprintf('Combining paths for B\n');
multi_f(B,sensors,endsec);
fprintf('Starting to combine sources A and B\n');
comb_f(A,B, sensors, endsec);
fprintf('Starting to combine data seconds\n');
sigcomb2_f(endsec);
fprintf('Starting simulation of noise with respect to source A\n');
clear a_speed;
a_speed=0;
sim62(A,a_track,a_speed,endsec);
fprintf('Starting to combine noise data seconds\n');
noisecomb_f(sensors, endsec);
fprintf('Starting to add noise to received signals\n');
clear
load noise1
load recsig1
recsig=recsig+noise;
save recsignoise1 recsig

clear
load noise2
load recsig2
recsig=recsig+noise;
save recsignoise2 recsig
clear
% end of signalgener22.m
clear
% end of signalgener22.m

20. File name: signalgener22d.m
% SIGNALGENER22D: Signal simulation for two sources with two receivers
% in a single path environment

fprintf('Starting simulation of source A
');
sim62(A,a_track,a_speed,endsec);
fprintf('Starting simulation of source B\n');
sim62(B,b_track,b_speed,endsec);
setBpower_f(sensors, endsec);
fprintf('Starting to combine sources A and B\n');
comb_f(A,B, sensors, endsec);
fprintf('Starting to combine data seconds\n');
sigcomb2_f(endsec);
fprintf('Starting to combine sources A and B\n');
clear a_speed;
aspeed=0;
sim62(A,a_track,a_speed,endsec);
fprintf('Starting to combine noise data seconds\n');
noisecomb_f(sensors, endsec);
fprintf('Starting to add noise to received signals\n');
clear
load noisel
load recsig1
recsig=recsig+noise;
save recsignoisel recsig
clear
load noise2
load recsig2
recsig=recsig+noise;
save recsignoise2 recsig
clear
% end of signalgener22d.m

21. File name: sim62.m
% SIM62: simulates path signals for a moving source with two receivers
% in a single path environment using sinx/x interpolation and store data
% in disk files one second for each file

function sim62(src, track_choice, speed, secmax)
tailpad_g = 600;
sampling_freq_g = 8000; % sampling frequency
sumpts = 100; % number of points +/- used in sinx/x summation
Tail = zeros(tailpad_g,1); % static pad space to clear Source
% filter edge effects
% Get more than we'll need--much easier than always checking whether
% there is enough data in the buffer.
needed = 1.1 * sampling_freq_g + sumpts;
sensor1 = 1; % sensor initialization
sensor2 = 2;
tick= 1; % send first data at T=0
if (src ==1),
    prefix1 = 'A1_';
    prefix2 = 'A2_';
    A_Trace = []; % save A_Trace A_Trace
else
    prefix1 = 'B1_';
    prefix2 = 'B2_';
    B_Trace = []; % save B_Trace B_Trace
end
BUFFEX1 = zeros(24000,3);
bufend1=0;
BUFFER2 = zeros(24000,3);  
bufend2=0;  
[PATH1,PATH2,Tail,VELO]=gener62_f(tick,src,track_choice,speed,Tail,VELO);  
% Save arrivals for use as tdoa at T=0;  
if src==1,  
   dai(tick) = PATH1(1,1);  
   da2(tick) = PATH2(1,1);  
else  
   db1(tick) = PATH1(1,1);  
   db2 (tick)= PATH2(1,1);  
end  
tick = tick+1;  
[BUFFER1, bufend1] = bufload_f(BUFFER1, bufend1, PATH1);  
[BUFFER2, bufend2] = bufload_f(BUFFER2, bufend2, PATH2);  
[BUFFER1, bufend1, secnum1] = align_f(BUFFER1, bufend1);  
[BUFFER2, bufend2, secnum2] = align_f(BUFFER2, bufend2);  
% Prepend an overlap of sumpts  
BUFFER1 = [zeros(sumpts,3);BUFFER1];  
BUFFER2 = [zeros(sumpts,3);BUFFER2];  
bufend1 = bufend1 + sumpts;  
bufend2 = bufend2 + sumpts;  
bufstart1 = sumpts +1;  
bufstart2 = sumpts +1;  
while (secnum1 <= secmax & secnum2 <= secmax)  
   while (bufend1 < needed | bufend2 < needed),  
      [PATH1,PATH2,Tail,VEL0]=gener62_f(tick,src,track_choice,speed,Tail,VEL0);  
      if src==1,  
         dai(tick) = PATH1(1,1); da2(tick) = PATH2(1,1);  
      else  
         db1(tick) = PATH1(1,1);  
      end  
      tick = tick+1;  
      [BUFFER1, bufend1] = bufload_f(BUFFER1, bufend1, PATH1);  
      [BUFFER2, bufend2] = bufload_f(BUFFER2, bufend2, PATH2);  
   end  
   while (bufend1 >= needed),  
      [RESAMP1,BUFFER1,bufend1] =...  
      resamp6a_f(BUFFER1,bufend1,secnum1,sensor1,sumpts);  
      fprintf('saving Sensor1 Second Number %2.0f\n',secnum1);  
      savesub6_f(RESAMP1, secnum1, prefix1);  
      secnum1 = secnum1 + 1;  
      end  
   while (bufend2 >= needed),  
      [RESAMP2,BUFFER2,bufend2] = resamp6a_f(BUFFER2,...  
      bufend2,secnum2,sensor2,sumpts);  
      fprintf('saving Sensor2 Second Number %2.0f\n',secnum2);  
      savesub6_f(RESAMP2, secnum2, prefix2);  
      secnum2 = secnum2 + 1;  
      end  
   end  
if (src ==1),  
   A_VELO = VELO; % save source velocity and location information  
   save A_VELO A_VELO;  
   save Adelay dai da2  
else  
   B_VELO = VELO;  
   save B_VELO B_VELO;  
   save Bdelay db1 db2  
end  
% end of sim62.m  

22. File name: sim6s2.m  
% SIM6S2: simulates path signals for a moving source with two  
% receivers in a multipath environment using sinx/x interpolation  
% and store data in disk files one second for each file
function sim6s2(src, track_choice, speed, secmax)
    tailpad_g = 600;
    sampling_freq_g = 8000;
    sumpts = 100; % number of points +/- used in sinx/x summation
    Tail = zeros(tailpad_g,1); % static pad space to clear Source filter
    % edge effects
    % Get more than we'll need--much easier than always checking whether
    % there is enough data in the buffer.
    needed = 1.1 * sampling_freq_g + sumpts;
    sensor1 = 1; % sensopr initialization
    sensor2 = 2;
    tick=1; % send first data at T=0
    if (src ==1),
        prefix1D = 'A1D_';
        prefix1S = 'A1S_';
        prefix2D = 'A2D_';
        prefix2S = 'A2S_';
        A_Trace = [];
        save A_Trace A_Trace
    else
        prefix1D = 'B1D_';
        prefix1S = 'B1S_';
        prefix2D = 'B2D_';
        prefix2S = 'B2S_';
        B_Trace = [];
        save B_Trace B_Trace
    end
    BUFFER1D = zeros(32000,3);
    BUFFER2D = zeros(32000,3);
    BUFFER1S = zeros(32000,3);
    BUFFER2S = zeros(32000,3);
    bufend2D=0;
    bufend1D=0;
    bufend2S=0;
    bufend1S=0;
    [PATH1D,PATH2D,PATH1S,PATH2S,Tail,VEL0] = ...
    gener6s2_f(tick, src, track_choice, speed, Tail,VEL0);
    % Save arrivals for use as tdoa at T=0;
    if src==1,
        da1(tick,:) = [PATH1D(1,1) PATH1S(1,1)];
        da2(tick,:) = [PATH2D(1,1) PATH2S(1,1)];
    else
        db1(tick,:) = [PATH1D(1,1) PATH1S(1,1)];
        db2(tick,:) = [PATH2D(1,1) PATH2S(1,1)];
    end
    tick = tick+1;
    [BUFFER1D, bufend1D] = bufload_f(BUFFER1D, bufend1D, PATH1D);
    [BUFFER2D, bufend2D] = bufload_f(BUFFER2D, bufend2D, PATH2D);
    [BUFFER1S, bufend1S] = bufload_f(BUFFER1S, bufend1S, PATH1S);
    [BUFFER2S, bufend2S] = bufload_f(BUFFER2S, bufend2S, PATH2S);
    [BUFFER1D, bufend1D, secum1D] = align_f(BUFFER1D, bufend1D);
    [BUFFER2D, bufend2D, secum2D] = align_f(BUFFER2D, bufend2D);
    [BUFFER1S, bufend1S, secum1S] = align_f(BUFFER1S, bufend1S);
    [BUFFER2S, bufend2S, secum2S] = align_f(BUFFER2S, bufend2S);
    % Prepend an overlap of sumpts
    BUFFER1D = [zeros(sumpts,3);BUFFER1D];
    BUFFER2D = [zeros(sumpts,3);BUFFER2D];
    BUFFER1S = [zeros(sumpts,3);BUFFER1S];
    BUFFER2S = [zeros(sumpts,3);BUFFER2S];
    bufend1D = bufend1D + sumpts;
    bufend2D = bufend2D + sumpts;
    bufend1S = bufend1S + sumpts;
    bufend2S = bufend2S + sumpts;
    bufstart1D = sumpts+1;
    bufstart2D = sumpts+1;
    bufstart1S = sumpts+1;
    bufstart2S = sumpts+1;
    while (secnum1D <= secmax & secnum2D <= secmax & ...
while (bufendiD < needed I bufend2D < needed I ...
    bufendiS < needed I bufend2S < needed ),
    [PATH1D, PATH2D, PATH1S, PATH2S, Tail, VEL0] = ...
    generateSD(s(tick, src, track_choice, speed, Tail, VEL0);
    if src==1,
        da1(tick,:) = [PATH1D(1,1) PATH1S(1,1)];
        da2(tick,:) = [PATH2D(1,1) PATH2S(1,1)];
    else
        db1(tick,:) = [PATH1D(1,1) PATH1S(1,1)];
        db2(tick,:) = [PATH2D(1,1) PATH2S(1,1)];
    end
    tick = tick+1;
    [BUFFER1D, bufendiD] = bufload_f(BUFFER1D, bufendiD, PATH1D);
    [BUFFER2D, bufend2D] = bufload_f(BUFFER2D, bufend2D, PATH2D);
    [BUFFER1S, bufendiS] = bufload_f(BUFFER1S, bufendiS, PATH1S);
    [BUFFER2S, bufend2S] = bufload_f(BUFFER2S, bufend2S, PATH2S);
end
[BUFFER1D, bufendiD] = bufload_f(BUFFER1D, bufendiD, PATH1D);
[BUFFER2D, bufend2D] = bufload_f(BUFFER2D, bufend2D, PATH2D);
[BUFFER1S, bufendiS] = bufload_f(BUFFER1S, bufendiS, PATH1S);
[BUFFER2S, bufend2S] = bufload_f(BUFFER2S, bufend2S, PATH2S);
% Prepend an overlap of sumpts
BUFFER1D = [zeros(sumpts,3);BUFFER1D];
BUFFER2D = [zeros(sumpts,3);BUFFER2D];
BUFFER1S = [zeros(sumpts,3);BUFFER1S];
BUFFER2S = [zeros(sumpts,3);BUFFER2S];
bufendiD = bufendiD + sumpts;
bufend2D = bufend2D + sumpts;
bufendiS = bufendiS + sumpts;
bufend2S = bufend2S + sumpts;
bufstart1D = sumpts +1;
bufstart2D = sumpts +1;
bufstart1S = sumpts +1;
bufstart2S = sumpts +1;
while (secnum1D <= secmax & secnum2D <= secmax & ...
    secnumIS <= secmax & secnum2S <= secmax )
    while (bufendiD < needed I bufend2D < needed I ...
        bufendiS < needed I bufend2S < needed ),
        [PATH1D, PATH2D, PATH1S, PATH2S, Tail, VEL0] = ...
        generateSD(s(tick, src, track_choice, speed, Tail, VEL0);
        if src==1,
            da1(tick,:) = [PATH1D(1,1) PATH1S(1,1)];
            da2(tick,:) = [PATH2D(1,1) PATH2S(1,1)];
        else
            db1(tick,:) = [PATH1D(1,1) PATH1S(1,1)];
            db2(tick,:) = [PATH2D(1,1) PATH2S(1,1)];
        end
        tick = tick+1;
        [BUFFER1D, bufendiD] = bufload_f(BUFFER1D, bufendiD, PATH1D);
        [BUFFER2D, bufend2D] = bufload_f(BUFFER2D, bufend2D, PATH2D);
        [BUFFER1S, bufendiS] = bufload_f(BUFFER1S, bufendiS, PATH1S);
        [BUFFER2S, bufend2S] = bufload_f(BUFFER2S, bufend2S, PATH2S);
end
while (bufendiD >= needed),
    [RESAMP1D,BUFFER1D,bufendiD] = resamp6a_f(BUFFER1D, ...
    bufendiD,secnum1D,sensor1,sumpts);
    fprintf('saving Sensorl Direct Second Number %2.0f\n',secnum1D);
    savesubs_f(RESAMP1D, secnum1D, prefix1D);
    secnum1D = secnum1D + 1;
end
while (bufend2D >= needed),
    [RESAMP2D,BUFFER2D,bufend2D] = resamp6a_f(BUFFER2D, ...
    bufend2D,secnum2D,sensor2,sumpts);
fprintf('saving Sensor2 Direct Second Number %2.0f
',secnum2D);
savesub6_f(RESAMP2D, secnum2D, prefix2D);
secnum2D = secnum2D + 1;
end

while (bufend1S >= needed),
    [RESAMP1S,BUFFER1S,bufend1S] = resamp6a_f(BUFFER1S, ...
bufend1S,secnum1S,sensor1,sumpts);
    fprintf('saving Sensor1 Surface Second Number %2.0f
',secnum1S);
savesub6_f(RESAMP1S, secnum1S, prefix1S);
secnum1S = secnum1S + 1;
end

while (bufend2S >= needed),
    [RESAMP2S,BUFFER2S,bufend2S] = resamp6a_f(BUFFER2S, ...
bufend2S,secnum2S,sensor2,sumpts);
    fprintf('saving Sensor2 Surface Second Number %2.0f
',secnum2S);
savesub6_f(RESAMP2S, secnum2S, prefix2S);
secnum2S = secnum2S + 1;
end

if (src ==1),
    A_VELO = VELO;
    save A_VELO A_VELO;
    save Adelay da1 da2
else
    B_VELO = VELO;
    save B_VELO B_VELO;
    save Bdelay db1 db2
end

23. File name: source_2.m
% SOURCE: function creates column vector with signal points
% Usage: Source = source2_f(Timebase, FILTER, filter_energy)
% where Timebase is column vector with sample times (in seq ??)
% level is approx. 1 watt after filtering (at 1 yard?)
% generate a sequence such that after filtering the signal power
% is 1 watt. Remember Tail sequence to preload next time,
% to avoid filter edge effects

function [Source,Tail]=...
source_2(Timebase,FILTER,filter_energy,testflag,oldTail)
samp_freq_4x = 32e3; % 4 times the sampling frequency
% Extra signal samples for interpolation
tailpad_g = 600;
if exist('randseed.mat')==2,
    load randseed;
else
    randseed=0;
end
randn('seed',randseed);
if testflag==1,
    Sequence = sin(2*pi*0.25*Timebase/samp_freq_4x) ./ sqrt(filter_energy);
else
    Sequence = randn(length(Timebase),1) ./ sqrt(filter_energy);
end
randseed = randn('seed');
save randseed randseed
% Now prefix new Seq with tail of last Seq
Sequence = [oldTail;Sequence];
% Save Tail for next time
Slen = length(Sequence);
Tail = Sequence(Slen-tailpad_g+1:Slen);
A = FILTER(1,:);
B = FILTER(2,:);
Source = filter(B,A,Sequence);
% discard edge effect at front.
if testflag==1,  
  save testsig Source  
end  
% end of source_2.m

24. File name: track5_f.m
% TRACK_F: creates a matrix of source spatial coordinates at
% sampling intervals
% Usage: TRACK = track5_f(Timebase, speed, choice)
% where speed is scalar source velocity in metres/second
% choice indicates which of several choices of track are desired.

function TRACK = track5_f(Timebase, speed, choice, freq)

x_comp = 1;  % index for x component
y_comp = 2;  % index for y component
z_comp = 3;  % index for z component
if choice==1,  % linear test track, constant speed
  Init_pos = [50 50 500];  % initial position
  Direction = [1/sqrt(2) 1/sqrt(2) 0];  % unit vector
elseif choice==2,  % linear test track, constant speed
  Init_pos = [500 500 500];  % initial position
  Direction = [-1/sqrt(2) -1/sqrt(2) 0];  % unit vector
elseif choice==3  % linear track, constant speed
  Init_pos = [1000 1000 500];  % initial position
  Direction = [1 0 0];  % unit vector
elseif choice==4  % linear track, constant speed
  Init_pos = [0 0 200];  % initial position
  Direction = [1 0 0];  % unit vector
elseif choice==5  % linear track, constant speed
  Init_pos = [500 2000 200];  % initial position
  Direction = [0 1 0];  % unit vector
elseif choice==6  % linear track, constant speed
  Init_pos = [500 2500 200];  % initial position
  Direction = [0 -1 0];  % unit vector
elseif choice==7  % linear track, constant speed
  Init_pos = [0 3500 200];  % initial position
  Direction = [1 0 0];  % unit vector
elseif choice==8  % linear track, constant speed
  Init_pos = [-2000 4000 500];  % initial position
  Direction = [1 0 0];  % unit vector
elseif choice==9  % linear track, constant speed
  Init_pos = [-2000 300 400];  % initial position
  Direction = [1 0 0];  % unit vector
elseif choice==10  % linear track, constant speed
  Init_pos = [-800 300 400];  % initial position
  Direction = [1 0 0];  % unit vector
else  % acceleration over linear track
  disp('Acceleration of source not implemented\n')
  break;
end

Sampletimes = Timebase ./ freq;
X = Init_pos(x_comp) + (Direction(x_comp) * speed) .* Sampletimes;
\[ Y = \text{Init} \_\text{pos}(y\_\text{comp}) + (\text{Direction}(y\_\text{comp}) \times \text{speed}) \times \text{Sampletimes}; \]
\[ Z = \text{Init} \_\text{pos}(z\_\text{comp}) + (\text{Direction}(z\_\text{comp}) \times \text{speed}) \times \text{Sampletimes}; \]
\[ \text{TRACK} = [X \ Y \ Z]; \]

% end of track5_f.m

25. File name: velo5a_f.m

% VELO5A_F: for each point of TRACK, compute closing speed
% (delay changing rate)
% Usage: Velo = velo5_f(Sensor_pos, TRACK, speed, choice, path_type);
% where Velo = [min mean max]
% rate change of delay. This program
% is much like track_f and also much like channel_f, but needs parts
% of both. To keep conceptually simple--keep separate.

\noindent
\textbf{function} Velo = velo5a_f(Sensor_pos, TRACK, speed, choice, path_type)
\textbf{sound} \_\text{speed} \_\text{g} = 1500; % sound speed underwater
\textbf{direct} \_\text{path} \_\text{g} = 1; % index for paths
\textbf{surface} \_\text{path} \_\text{g} = 2;
\textbf{x} \_\text{comp} = 1; % index for x component
\textbf{y} \_\text{comp} = 2; % index for y component
\textbf{z} \_\text{comp} = 3; % index for z component

% Determine the RELATIVE path between source and sensor
\text{REL} \_\text{TRACK}(:, \text{x} \_\text{comp}) = TRACK(:, \text{x} \_\text{comp}) - Sensor_pos(\text{x} \_\text{comp});
\text{REL} \_\text{TRACK}(:, \text{y} \_\text{comp}) = TRACK(:, \text{y} \_\text{comp}) - Sensor_pos(\text{y} \_\text{comp});
\textbf{if} path \_\text{type} == \text{direct} \_\text{path} \_\text{g};
\text{REL} \_\text{TRACK}(:, \text{z} \_\text{comp}) = TRACK(:, \text{z} \_\text{comp}) - Sensor_pos(\text{z} \_\text{comp});
\textbf{elseif} path \_\text{type} == \text{surface} \_\text{path} \_\text{g},
\text{REL} \_\text{TRACK}(:, \text{z} \_\text{comp}) = TRACK(:, \text{z} \_\text{comp}) - (-1 \times Sensor_pos(\text{z} \_\text{comp}));
\textbf{else}
\text{disp}'Unknown path\_type in Channel_f'\textbf{end}

% Find the Euclidian distances between source and sensor
\text{Euclid} = \sqrt{\text{sum}((\text{REL} \_\text{TRACK} - \text{2}))'};
\textbf{if} choice == 1, % linear test track, constant speed
\text{Init} \_\text{pos} = [50 \ 50 \ 500]; % initial position
\text{Direction} = [\text{1}/\sqrt{2} \ \text{1}/\sqrt{2} \ 0]; % unit vector
\text{else} choice == 2, % linear test track, constant speed
\text{Init} \_\text{pos} = [500 \ 500 \ 500]; % initial position
\text{Direction} = [-\text{1}/\sqrt{2} \ -\text{1}/\sqrt{2} \ 0]; % unit vector
\textbf{elseif} choice == 3 % linear track, constant speed
\text{Init} \_\text{pos} = [1000 \ 1000 \ 500]; % initial position
\text{Direction} = [-\text{1} \ \text{0} \ \text{0}]; % unit vector
\textbf{elseif} choice == 4 % linear track, constant speed
\text{Init} \_\text{pos} = [0 \ 0 \ 200]; % initial position
\text{Direction} = [\text{1} \ \text{0} \ \text{0}]; % unit vector
\textbf{elseif} choice == 5 % linear track, constant speed
\text{Init} \_\text{pos} = [1000 \ 4000 \ 200]; % initial position
\text{Direction} = [-\text{1} \ \text{0} \ \text{0}]; % unit vector
\textbf{elseif} choice == 6 % linear track, constant speed
\text{Init} \_\text{pos} = [1000 \ 3000 \ 200]; % initial position
\text{Direction} = [\text{1} \ \text{0} \ \text{0}]; % unit vector
\textbf{elseif} choice == 7 % linear track, constant speed
\text{Init} \_\text{pos} = [500 \ 2000 \ 200]; % initial position
\text{Direction} = [\text{0} \ \text{1} \ \text{0}]; % unit vector
\textbf{elseif} choice == 8 % linear track, constant speed
\text{Init} \_\text{pos} = [500 \ 2500 \ 200]; % initial position
\text{Direction} = [\text{0} \ -\text{1} \ \text{0}]; % unit vector
\textbf{elseif} choice == 9 % linear track, constant speed
\text{Init} \_\text{pos} = [0 \ 3500 \ 200]; % initial position
\text{Direction} = [\text{1} \ \text{0} \ \text{0}]; % unit vector
\textbf{elseif} choice == 10 % linear track, constant speed
\text{Init} \_\text{pos} = [0 \ 2000 \ 200]; % initial position
\text{Direction} = [-\text{1} \ \text{0} \ \text{0}]; % unit vector
\textbf{elseif} choice == 11 % linear track, constant speed
\text{Init} \_\text{pos} = [-\text{500} \ 4000 \ 200]; % initial position
\text{Direction} = [\text{1} \ \text{0} \ \text{0}]; % unit vector
\textbf{elseif} choice == 12 % linear track, constant speed
Init_pos = [500 3500 200]; % initial position
Direction = [-1 0 0]; % unit vector
elseif choice==13 % linear track, constant speed
    Init_pos = [-2000 2000 200]; % initial position
    Direction = [1 0 0]; % unit vector
elseif choice==14 % linear track, constant speed
    Init_pos = [-2000 400 500]; % initial position
    Direction = [1 0 0]; % unit vector
elseif choice==15 % linear track, constant speed
    Init_pos = [-800 300 400]; % initial position
    Direction = [1 0 0]; % unit vector
else % acceleration over linear track
    disp('Acceleration of source not implemented')
    break;
end

% Make unit vector in direction from sensor to source
DIREC(:,1) = REL_TRACK(:,1) ./ Euclid;
DIREC(:,2) = REL_TRACK(:,2) ./ Euclid;
DIREC(:,3) = REL_TRACE(:,3) ./ Euclid;
% Project the source speed onto the direction vector (dot product)
Proj = speed * (Direction * DIREC')';
% Return min, mean and max rate change of delay (Proj/speed_of_sound)
Velo = [Proj(1) mean(Proj) max(Proj)]/sound_speed_g;
% end of veloSa.f.m

A.2 Implementation of the TDOA Estimation Algorithm

This is the implementation of the TDOA estimation algorithm in MATLAB programming. These programs estimate the TDOAs of the received signals in the cross-correlating and autocorrelating function.

A.2.1 Program Structure

The program structure of implementing the TDOA estimation in MATLAB can be shown by a tree diagram of the main program and subprograms in Figure A.2. These programs estimate the TDOAs by cross-correlating the received signals. A similar tree diagram for the programs estimating the TDOAs by autocorrelating the received signals is shown in Figure A.3.

A.2.2 Usage

Before executing the programs, the range of differential time delay and relative time companding for the search of TDOA estimates have to be specified in the
Figure A.2: Tree diagram of the programs in implementing the TDOA estimation using cross-correlation

Figure A.3: Tree diagram of the programs in implementing the TDOA estimation using autocorrelation
main program `peaksearch.m`, for estimating cross-correlation TDOAs, and `autopeaksearch.m`, for estimating autocorrelating TDOAs.

- The range of differential time delay $\tau$ can be specified by entering the value for `delay_range` in the format:

  \[
  \text{[Start value : Step size : Stop value]}
  \]

  Note that the value of differential time delay is in unit of sampling time interval $T_s p$, i.e., $T_s p = 1/f_s \text{second}$ and $f_s = 8000$ as default value for sampling frequency.

- The range of relative time companding $\beta$ can be specified by entering the value of `beta_range` in the format:

  \[
  \text{[Start value : Step size : Stop value]}
  \]

  Note the relative time companding is the ratio of the time delay changing rate of the the two signals.

Some other parameters are:

- Short-Time intervals $ts$ and $td$: The default value for $ts$ is $200T_s p$ and for $td$ is $100T_s p$ which can be changed in the file `peak_locate.m` and `peaksearch.m` (or `autopeaksearch.m` for estimating from the autocorrelation function)

- Number of accumulated correlograms $n$: The default value for $n$ is 200 which can be changed in the format $n = \text{start value : stop value}$, in the file `cross_corr.m` and $n - 1$ in the file `shift_add.m`

After entering necessary parameters, the programs to estimate the TDOAs from cross-correlation can be executed by typing

```
peaksearch (Enter)
```

and to estimate the TDOAs from autocorrelation

```
autopeaksearch (Enter)
```

on the MATLAB command window.

### A.2.3 Output Format

The output of the programs is a three column matrix in the format:

\[
[\beta, \tau, \text{Row number}]
\]
The first column, the $\beta$ column, stores the value of the relative time companding. The second column, the $\tau$ column, stores the value of the differential time delay. The third column stores the row number of the matrix. This row number is for the purpose of marking the differential time delays to be used in the registration process. The value of the differential time delay will be given in unit of sampling time interval $T_s p$.

The output generated from the programs includes different set of the TDOA estimation corresponding to different intervals of the time observation periods with known time increment.

The nomenclature for the output files are:
- *auto10* the TDOA estimates from autocorrelation for received signal at sensor 1, zeroth time observation period
- *auto11* the TDOA estimates from autocorrelation for received signal at sensor 1, first time observation period
- *auto12* the TDOA estimates from autocorrelation for received signal at sensor 1, second time observation period
- *delaypeak0* The TDOA estimates from cross-correlation for the zeroth time observation period
- *delaypeak1* the TDOA estimates from cross-correlation for the first time observation period
- *delaypeak2* the TDOA estimates from cross-correlation for the second time observation period

### A.2.4 Program Code

The following programs are written for UNIX environment. Documentation is included in the file with a % sign at the beginning of the first line.

1. File name: auto_corr.m

```matlab
%% AUTO_CORR: autocorrelates short time intervals of the received
%% signal for a given delay tau and relative time companding
%% beta and stores the resulting correlograms in a matrix.
```
function auto_corr(r1,td,ts,tau)
    si = r1(48000-2*ts+1 : 48000); % delaying by tau
    s2 = r1(48000-tau-ts+1 : 48000-tau); % delaying by tau
    xcorr12 = xcorr(si,s2);
    dt=xcorr12(ts+1:3*ts-1); % twice ts in the first correlation
    % to avoid edge effect
    dt=dt/max(abs(dt));
    xc(1,:)=dt;
    clear xcorr12;
    for n = 2:200;
        si = r1(48000-((n-1)*td+ts)+1: 48000-(n-1)*td); % delaying by tau and
        s2 = r1(48000-tau+(n-1)*td+1: 48000-tau-(n-1)*td); % take ts
        xcorrl2=xcorr(sl,s2); % cross-correlating short time interval ts
        xcorr12=xcorr12/max(abs(xcorr12));
        xc(n,:)=xcorr12;
    end
    save xc xc; % save resulting correlograms to file
end

2. File name: autompeaksearch.m

% AUTOPEAKSEARCH: This is the main program which estimates the TDOAs % resulting from autocorrelating the received signals for different % time observation periods with known time increment

    td = 200; % length of short time interval td (unit=1/8000 sec)
    ts = 100; % length of short time segment ts for correlation
    % Estimate the TDOAs from autocorrelation for received signal at sensor 1
    clear;
    for p=0:10; % p=number of time observation periods
        load recsig1
        r1 = recsig(1+800*(p):48000+800*(p)); % observation time period = 6 seconds
        clear recsig;
        delay_range=[400:26:1600]; % range of tau for TDOA search
        beta_range=[.9985:0.0001:1.0005]; % range of beta for TDOA search
        autotot_search(delay_range,beta_range,r1,td,ts);
        load peak_location
        1=length(peak_location(:,1));
        count1=0;
        str=eval(['autoi , ,int2str(p),"\"']);
        % Searching for correct TDOA peaks
        for k=2:1-1;
            peakamp1=abs(peak_location(k-1,3));
            peakamp2=abs(peak_location(k,3));
            peakamp3=abs(peak_location(k+1,3));
            timediff1=abs(peak_location(k,2)-peak_location(k-1,2));
            timediff2=abs(peak_location(k,2)-peak_location(k+1,2));
            if peakamp2>peakamp1 & peakamp2>peakamp3 & timediff1<10 & timediff2>10
                count1=count1+1;
                auto1(count1,1:2)=peak_location(k,1:2);
                auto1(count1,3)=count1;
            elseif peakamp2<peakamp1 & peakamp2<peakamp3 & timediff1<10 & timediff2<10
                count1=count1+1;
                auto1(count1,1:2)=peak_location(k,1:2);
                auto1(count1,3)=count1;
            elseif peakamp2>peakamp1 & peakamp2>peakamp3 & timediff1>10 & timediff2<10
                count1=count1+1;
                auto1(count1,1:2)=peak_location(k,1:2);
                auto1(count1,3)=count1;
            end
        end
        eval(['"save ',str,' autos"']) % save TDOA estimates to file
        clear peak_location;
        clear betaTau_peak;
        clear auto1;
        clear r1;
    % Repeat the whole process to estimate the TDOAs from autocorrelation
for received signal at sensor 2
load recsig2
r2 = recsig(1+800*p:48000+800*p);
clear recsig;
delay_range=[400:25:1600];
beta_range=[.9985:0.0001:1.0005];
autotot_search(delay_range,beta_range,r2,td,ts);
load peak_location
1=length(peak_location(:,1));
count1=0;
str=eval(['"auto2 , ,int2str(p),''''])
% Searching for correct TDOA peaks
for k=2:1-1;
peakampl=abs(peak_location(k-1,3));
peakamp2=abs(peak_location(k,3));
peakamp3=abs(peak_location(k+1,3));
timediff1=abs(peak_location(k,2)-peak_location(k-1,2));
timediff2=abs(peak_location(k,2)-peak_location(k+1,2));
if peakamp2>peakamp1 & peakamp2>peakamp3
count1=count1+1;
auto2(count1,1:2)=peak_location(k,1:2);
auto2(count1,3)=count1;
elseif peakamp2>peakamp1 & peakamp2<peakamp3 & timediff1<10 & timediff2>10
count1=count1+1;
auto2(count1,1:2)=peak_location(k,1:2);
auto2(count1,3)=count1;
else if peakamp2<peakamp1 & peakamp2>peakamp3 & timediff1<10 & timediff2>10
count1=count1+1;
auto2(count1,1:2)=peak_location(k,1:2);
auto2(count1,3)=count1;
end
end
eval(['"save ',str,'
% save TDOA estimates to file
clear peak_location;
clear beta_tau_peak;
clear auto2;
clear r2;
end
% end of autopeaksearch.m

3. File name: autotot_search.m
% AUTOTOT_SEARCH: locates all the peaks from the autocorrelation of two received signals. This function returns the peaks value with differential time delay tau and relative time companding beta corresponding to each peak for given range of delay and beta. A passive ambiguity surface can be formed with the returning values.

function autotot_search(delay_range,beta_range,r1,td,ts)
delaysteps=length(delay_range); % step size for tau and beta
betasteps=length(beta_range);
for i=1:delaysteps,
tau=delay_range(i);
auto_corr(r1,td,tc,tau); % autocorrelating data sequence
load xc
peak_locate(xc,beta_range,tau); % locates the peaks
load beta_tau_peak
[y,1]=max(abs(beta_tau_peak(:,3)));
peak_location(1,:)=beta_tau_peak(1,:);
end
save peak_location peak_location % save peak location to file
% end of autotot_search.m

4. File name: cross_corr.m
% CROSS_CORR: cross-correlates short time intervals of the two received signals for a given delay tau and relative time companding beta and stores in a matrix.
function cross_corr(r1, r2, td, ts, tau)
    sl = r1(48000-2*tc+1 : 48000);
    s2 = r2(48000-tau-ts+1 : 48000-tau);
    xcorr12 = xcorr(sl, s2);
    dt=xcorr12(ts+1:3*ts-1); \% twice the short time interval tc in the first
    \% correlation to avoid edge effect
    dt=dt/max(abs(dt));
    xc(1,:)=dt;
    clear xcorrl2;
    for n = 2:200;
        sl = r1(48000-((n-1)*td+ts)+1: 48000-((n-1)*td)); \% delaying by tau and
        s2 = r2(48000-(tau+ts+(n-1)*td)+1: 48000-tau-(n-1)*td); \% take ts
        xcorr12=xcorr(sl, s2); \% cross-correlating short time interval ts
        xcorr12=xcorr12/max(abs(xcorr12));
        xc(n,:)=xcorr12;
    end
    save xc xc; \% save resulting correlograms to file
    end of cross_corr.m

5. File name: peak_locate.m
\% PEAK_LOCATE: searches all peaks in the correlograms for a tau
\% value within a beta range. It will return a three columns matrix
\% recording the peak value with corresponding beta and tau value.

function peak_locate(xc, beta_range, tau)
    n=length(beta_range);
    td=200; \% length of short time interval td (unit= 1/8000 sec)
    tc=100; \% length of short time interval ts
    for i=1:n,
        beta=beta_range(i);
        shift_add(xc, beta, td, tc);
        load corrsum
        [y, l]=max(abs(corrsum));
        peak=corrsum(1);
        if y>30 \% set threshold value to reduce false peaks from noise
            if l<2
                beta_tau_peak(i,1)=beta; \% search for the peak in the correlogram
                beta_tau_peak(i,2)=tau+tc-1;
                beta_tau_peak(i,3)=peak;
                elseif l>198
                    beta_tau_peak(i,1)=beta;
                    beta_tau_peak(i,2)=tau-tc+1;
                    beta_tau_peak(i,3)=peak;
            else
                \% interpolating to find exact location of the peak in the correlogram
                delay=l+(corrsum(1+1)-corrsum(1-1))/(2*(corrsum(1+1)+2*peak-corrsum(1-1)));
                diff_delay=tc-delay;
                beta_tau_peak(i,1)=beta;
                beta_tau_peak(i,2)=tau+diff_delay;
                beta_tau_peak(i,3)=peak;
                end
            else
                beta_tau_peak(i,1)=beta;
                beta_tau_peak(i,2)=tau;
                beta_tau_peak(i,3)=0;
            end
        end
    end
    save beta_tau_peak beta_tau_peak \% save data to file
    end of peak_locate.m

6. File name: peaksearch.m
\% PEAKSEARCH: This the main program which estimates the TDOAs
\% resulting from cross-correlating the received signal for
\% different observation time periods with known time increment.

clear
td = 100; % length of short time interval td (unit=1/8000 sec)
ts = 100; % length of short time interval ts
for p=0:10;
load recsig1
r1 = recsig(1+p*(800):48000+p*(800)); % observation time period
load recsig2
r2 = recsig(1+p*(800):48000+p*(800)); % observation time period
clear recsig;
delay_range=[3100:25:5700]; % range of tau for the search
beta_range=[.9996:0.0001:1.0024]; % range of beta for the search
tot_search(delay_range,beta_range,r1,r2,td,ts);
load peak_location
l=length(peak_location(:,1));
coun+=0;
str=eval(['''delaypeak',int2str(p),'''']);
% Search for correct TDOA peaks
for k=2:1-1;
peakamp1=abs(peak_location(k-1,3));
peakamp2=abs(peak_location(k,3));
peakamp3=abs(peak_location(k+1,3));
timediff1=abs(peak_location(k,2)-peak_location(k-1,2));
timediff2=abs(peak_location(k,2)-peak_location(k+1,2));
if peakamp2>peakamp1 & peakamp2>peakamp3 & timediff1<10 & timediff2<10
count1=count1+1;
delaypeaka(count1,1:2)=peak_location(k,1:2);
delaypeaka(count1,1:3)=count1;
elseif peakamp2>peakamp1 & peakamp2>peakamp3 & timediff1<10 & timediff2>10
count1=count1+1;
delaypeaka(count1,1:2)=peak_location(k,1:2);
delaypeaka(count1,1:3)=count1;
elseif peakamp2>peakamp1 & peakamp2>peakamp3 & timediff1>10 & timediff2<10
count1=count1+1;
delaypeaka(count1,1:2)=peak_location(k,1:2);
delaypeaka(count1,1:3)=count1;
elseif peakamp2>peakamp1 & peakamp2>peakamp3 & timediff1>10 & timediff2>10
count1=count1+1;
delaypeaka(count1,1:2)=peak_location(k,1:2);
delaypeaka(count1,1:3)=count1;
end
% Arrange the TDOA estimates in an ascending order according to their
% values in time
nrofdelay1=length(delaypeaka(:,2))
for num1=nrofdelay1:-1:2;
nrofdelay2=length(delaypeaka(:,2));
value1=max(delaypeaka(:,2));
for num2=1:nrofdelay2;
value2=value1-delaypeaka(num2,2);
if value2 == 0
delaypeak(num1,:)=delaypeaka(num2,:);
delaypeaka(num2,:)=delaypeaka(num2,:)-delaypeaka(num2,:);
end
end
delaytemp(:,1)=nonzeros(delaypeaka(:,1))
delaytemp(:,2)=nonzeros(delaypeaka(:,2))
delaytemp(:,3)=nonzeros(delaypeaka(:,3))
clear delaypeaka
delaypeaka=delaytemp
clear delaytemp
delaypeak(1,:)=delaypeak(1,:);
eval(['''save ',str,' delaypeak']) % save data to file
% clear variables for the next search
clear peak_location
delete peak_location.mat
clear beta_tau_peak
delete beta_tau_peak.mat
clear delaypeak
end
% end of peaksearch.m

7. File name: shift_add.m
% SHIFT_ADD: time shift the correlograms with given beta value and
% sums them up to give the time average correlation
function shift_add(xc,beta,td,ts)
shiftstep = (beta-1)*td; % step size of time shifting
corrsum = xc(1,:);
for n=1:199;
    shift = round(n*shiftstep);
    if shiftstep>0 % if beta is positive shift forwards
        correlogram(1,1:shift)=zeros(1,shift);
        correlogram(1,shift+1:199)=xc(n+1,1:199-shift);
    elseif shift==0
        correlogram(1,1:199)=xc(n+1,1:199);
    else % if beta is negative shift backwards
        shift=abs(shift);
        correlogram(1,1:199-shift)=xc(n+1,shift+1:199);
        correlogram(1,199-shift+1:199)=zeros(1,shift);
    end
    corrsum=corrsum+correlogram;
end
save corrsum corrsum % save sum to file
% end of shift_add.m

8. File name: tot_search.m
% TOT_SEARCH: locates all the peaks from the resulting time average
% cross-correlation of two received signals. This function returns
% differential time delay tau and relative time companding beta
% corresponding for each peak in the correlograms for given range of
% tau and beta. A passive ambiguity surface can be formed with the
% returning values.
function tot_search(delay_range,beta_range,rl,r2,td,ts)
delaysteps=length(delay_range); % step size of tau in the search
for i=1:delaysteps,
delay=delay_range(i);
cross_corr(r1,r2,td,ts,delay);
end
load xc
peak_locate(xc,beta_range,delay); % peak location
load beta_tau_peak
[y,l]=max(abs(beta_tau_peak(:,3)));
peak_location(l,:)=beta_tau_peak(1,:);
end
save peak_location peak_location
% end of tot_search.m

A.3 Implementation of The TDOA Estimates Registration Algorithm

This is the implementation of the TDOA estimates registration algorithm in MATLAB programming. The programs will classify the TDOA estimates to the corresponding source and register them to the pair of path signals that produce each
TDOA estimate.

### A.3.1 Program Structure

The program structure for the implementation of the TDOA registration can be summarized by a tree diagram with the main program and subprograms as shown in Figure A.4.

![Tree diagram of the programs in implementing the TDOA estimates registration](image)

Figure A.4: Tree diagram of the programs in implementing the TDOA estimates registration

### A.3.2 Usage

The program can be executed simply by typing

```
delaycheck (Enter)
```

on the MATLAB command window.

### A.3.3 Output Format

The TDOA estimates registration output are given in a column matrix format with the associated path pair. For example

\[ d_{1d2} = \text{value in time} \]
In the case of missing peaks where the source TDOA registration cannot be determined exactly, possible results will be given.

### A.3.4 Program Code

The following programs are written for UNIX environment. Documentation is included in the file with a % sign at the beginning of the first line.

```matlab
% File name: delaycheck.m
% DELAYCHECK: this is the main program for the registration which
% classifies the TDOA estimates to the corresponding source and
% registers the TDOA estimates to the pair of path signals that
% produce each TDOA estimate

for num=1:10; % number of observation time periods
numstr=int2str(num);
str1 = eval(['''auto1'' numstr '.mat''']);
str2 = eval(['''auto2'' numstr '.mat''']);
strd = eval(['''delaypeak'' numstr '.mat''']);
if (exist(str1)==2)
eval(['''load '' str1'' ]); % check available TDOA estimates from
else
11=0;
end
if (exist(str2)==2)
eval(['''load '' str2'' ]); % check available TDOA estimates from
12=length(auto2(:,2)); % autocorrelation at sensor 2
else
12=0;
end
eval(['''load '' strd'' ]); % check available TDOA estimates from
13=length(delaypeak(:,2)); % cross-correlation

% Classification of the TDOA estimates to the corresponding source
if ( (11==2) & (12==2) ) % using available TDOA estimates to
check1 % decide the appropriate case
elseif ( (11==2) & (12==1) )
check2
elseif ( (11==2) & (12==0) )
check3
elseif ( (11==1) & (12==2) )
check4
elseif ( (11==0) & (12==2) )
check5
elseif ( (11==0) & (12==1) )
check6
else
fprintf('Insufficient data for TDOA registration
')
end
% Registration of the TDOA estimates to the pair of path signals
% that produce each TDOA estimate and printing correct results
```
if exist('sourcedelay')
    strs = eval(['''sourcedelay'' numstr ']])
    eval(['save ',strs,' sourcedelay'])
    numdelay=length(sourcedelay)/4;
    for m=1:numdelay;
        fprintf('Source detected with the TDOA Registration
')
        if ( sourcedelay(2+(m-1)*4,1)<0 ) & ( sourcedelay(3+(m-1)*4,1)<0 )
            fprintf('path pair dis2 = %1.0f
',sourcedelay(1+(m-1)*4,1)/8000)
            fprintf('path pair sis2 = %1.0f
',sourcedelay(2+(m-1)*4,1)/8000)
            fprintf('path pair did2 = %1.0f
',sourcedelay(3+(m-1)*4,1)/8000)
            fprintf('path pair sid2 = %1.0f
',sourcedelay(4+(m-1)*4,1)/8000)
        end
    end
    clear sourcedelay
    else
        fprintf('Possible TDOA Registrations for one source
')
        if ( sourcedelay(2+(m-1)*4,1)<0 ) & ( sourcedelay(3+(m-1)*4,1)<0 )
            fprintf('path pair dis2 = %1.0f
',sourcedelay(1+(m-1)*4,1)/8000)
            fprintf('path pair sis2 = %1.0f
',sourcedelay(2+(m-1)*4,1)/8000)
            fprintf('path pair did2 = %1.0f
',sourcedelay(3+(m-1)*4,1)/8000)
            fprintf('path pair sid2 = %1.0f
',sourcedelay(4+(m-1)*4,1)/8000)
        end
    end
    clear sourcedelay
elseif exist('source1')
    strs = eval(['''source1'' numstr ']])
    eval(['save ',strs,' source1'])
    numdelay=length(source1)/4;
    for m=1:numdelay;
        fprintf('Possible TDOA Registrations for one source
')
        if ( source1(2+(m-1)*4,1)<0 ) & ( source1(3+(m-1)*4,1)<0 )
            fprintf('path pair dis2 = %1.0f
',source1(1+(m-1)*4,1)/8000)
            fprintf('path pair sis2 = %1.0f
',source1(2+(m-1)*4,1)/8000)
            fprintf('path pair did2 = %1.0f
',source1(3+(m-1)*4,1)/8000)
            fprintf('path pair sid2 = %1.0f
',source1(4+(m-1)*4,1)/8000)
        end
    end
    clear source1
elseif exist('source2')
    strs = eval(['''source2'' numstr ']])
    eval(['save ',strs,' source2'])
    numdelay=length(source2)/4;
    for m=1:numdelay;
        fprintf('Possible TDOA Registrations for one source
')
        if ( source2(2+(m-1)*4,1)<0 ) & ( source2(3+(m-1)*4,1)<0 )
            fprintf('path pair dis2 = %1.0f
',source2(1+(m-1)*4,1)/8000)
            fprintf('path pair sis2 = %1.0f
',source2(2+(m-1)*4,1)/8000)
            fprintf('path pair did2 = %1.0f
',source2(3+(m-1)*4,1)/8000)
            fprintf('path pair sid2 = %1.0f
',source2(4+(m-1)*4,1)/8000)
        end
    end
    clear source2
elseif exist('source3') \nstrs = eval(['''source3' numstr '''])
eval(['''save ',strs, ' source3'''])
numdelay=length(source3)/4;
for m=1:numdelay;
    fprintf('++++++++++++++++++++++++W)
    fprintf('Possible TDOA Registrations for one source
    if ( source3(2+(m-1)*4,1)<0 ) & ( source3(3+(m-1)*4,1)<0 )
    fprintf('path pair dis2 = %1.0f\n',source3(1+(m-1)*4,1)/8000)
    fprintf('path pair sis2 = %1.0f\n',source3(2+(m-1)*4,1)/8000)
    fprintf('path pair did2 = %1.0f\n',source3(3+(m-1)*4,1)/8000)
    fprintf('path pair s1d2 = %1.0f\n',source3(4+(m-1)*4,1)/8000)
    fprintf(' 
')
    else
    fprintf('path pair dis2 = %1.0f\n',source3(1+(m-1)*4,1)/8000)
    fprintf('path pair did2 = %1.0f\n',source3(2+(m-1)*4,1)/8000)
    fprintf('path pair sis2 = %1.0f\n',source3(3+(m-1)*4,1)/8000)
    fprintf('path pair sld2 = %1.0f\n',source3(4+(m-1)*4,1)/8000)
    fprintf(' 
')
end
end
clear source3
elseif exist('source4')
strs = eval(['''source4' numstr '''])
eval(['''save ',strs, ' source4'''])
numdelay=length(source4)/4;
for m=1:numdelay;
    fprintf('++++++++++++++++++++++++
    fprintf('Possible TDOA Registrations for one source
    if ( source4(2+(m-1)*4,1)<0 ) & ( source4(3+(m-1)*4,1)<0 )
    fprintf('path pair dis2 = %1.0f\n',source4(1+(m-1)*4,1)/8000)
    fprintf('path pair did2 = %1.0f\n',source4(2+(m-1)*4,1)/8000)
    fprintf('path pair sis2 = %1.0f\n',source4(3+(m-1)*4,1)/8000)
    fprintf('path pair sld2 = %1.0f\n',source4(4+(m-1)*4,1)/8000)
    fprintf(' 
')
    else
    fprintf('path pair dis2 = %1.0f\n',source4(1+(m-1)*4,1)/8000)
    fprintf('path pair did2 = %1.0f\n',source4(2+(m-1)*4,1)/8000)
    fprintf('path pair sis2 = %1.0f\n',source4(3+(m-1)*4,1)/8000)
    fprintf('path pair sld2 = %1.0f\n',source4(4+(m-1)*4,1)/8000)
    fprintf(' 
')
end
end
clear source4
else
    fprintf('No source was detected\n')
end
% end of delaycheck.m

2. File name: check1.m
% CHECK1: this subprogram classifies the delay pairs sld2,d1s2 and
% did2,sis2 for the case when all autocorrelation delays are present
% and check for possibility of missing peak (calculate whenever possible)

suma1=auto1(1,2)+auto2(1,2);
suma2=auto1(1,2)+auto2(2,2);
suma3=auto1(2,2)+auto2(1,2);
suma4=auto1(2,2)+auto2(2,2);
for k=13:-1:2;
    for m=k-1:-1:1;
        diff1=abs(delaypeak(k,2)-delaypeak(m,2)-suma1);
        diff2=abs(delaypeak(k,2)-delaypeak(m,2)-suma2);
        diff3=abs(delaypeak(k,2)-delaypeak(m,2)-suma3);
        diff4=abs(delaypeak(k,2)-delaypeak(m,2)-suma4);
        if diff1<5
            fprintf('path pair dis2 = %1.0f\n',source3(1+(m-1)*4,1)/8000)
            fprintf('path pair sid2 = %1.0f\n',source3(2+(m-1)*4,1)/8000)
            fprintf(' 
')
        end
    end
end
% end of check1.m
sld2d1s2(1+(count1-1)*2,1:3)=delaypeak(m,:);
sld2d1s2(1+(count1-1)*2,4)=auto1(1,2);
sld2d1s2(2+(count1-1)*2,1:3)=delaypeak(k,:);
sld2d1s2(2+(count1-1)*2,4)=auto2(1,2);
else if diff2<5
  count1=count1+1;
sld2d1s2(1+(count1-1)*2,1:3)=delaypeak(m,:);
sld2d1s2(1+(count1-1)*2,4)=auto1(1,2);
sld2d1s2(2+(count1-1)*2,1:3)=delaypeak(k,:);
sld2d1s2(2+(count1-1)*2,4)=auto2(2,2);
else if diff3<5
  count1=count1+1;
sld2d1s2(1+(count1-1)*2,1:3)=delaypeak(m,:);
sld2d1s2(1+(count1-1)*2,4)=auto1(2,2);
sld2d1s2(2+(count1-1)*2,1:3)=delaypeak(k,:);
sld2d1s2(2+(count1-1)*2,4)=auto2(1,2);
else if diff4<5
  count1=count1+1;
sld2d1s2(1+(count1-1)*2,1:3)=delaypeak(m,:);
sld2d1s2(1+(count1-1)*2,4)=auto1(2,2);
sld2d1s2(2+(count1-1)*2,1:3)=delaypeak(k,:);
sld2d1s2(2+(count1-1)*2,4)=auto2(2,2);
end
suba11=-auto1(1,2)+auto2(1,2);
suba12=-auto1(1,2)+auto2(2,2);
suba21=-auto1(2,2)+auto2(1,2);
suba22=-auto1(2,2)+auto2(2,2);
diff5=abs(delaypeak(k,2)-delaypeak(m,2)-suba11);
diff6=abs(delaypeak(k,2)-delaypeak(m,2)-suba12);
diff7=abs(delaypeak(k,2)-delaypeak(m,2)-suba21);
diff8=abs(delaypeak(k,2)-delaypeak(m,2)-suba22);
% Check if the TDOA estimate pair are did2,s1s2
if ( (exist('s1d2d1s2')==1) k (exist('d1d2s1s2')==1) )
  num1=length(s1d2d1s2(:,2))/2;
  num2=length(d1d2s1s2(:,2))/2;
  templ=zeros(size(s1d2d1s2));
  temp2=zeros(size(d1d2s1s2));
  num3=0;
  for k=1:num1;
    for m=1:num2;
      % Classifying the TDOA estimate pairs to the corresponding source
      if ( (exist('s1d2d1s2')==1) k (exist('did2s1s2')==1) )
        num1=length(s1d2d1s2(:,2))/2;
        num2=length(d1d2s1s2(:,2))/2;
        templ=zeros(size(s1d2d1s2));
        temp2=zeros(size(d1d2s1s2));
        num3=0;
        for k=1:num1;
          for m=1:num2;
sum1 = abs(s1d2d1s2(1+2*(k-1),2)+s1d2d1s2(2+2*(k-1),2) ... -d1d2s1s2(1+2*(m-1),2)-d1d2s1s2(2+2*(m-1),2));
if sum1<5
    num3 = num3+1;
    templ(1+2*(k-1),:)=s1d2d1s2(1+2*(k-1),:);
    templ(2+2*(k-1),:)=s1d2d1s2(2+2*(k-1),:);
    temp2(1+2*(m-1),:)=d1d2s1s2(1+2*(m-1),:);
    temp2(2+2*(m-1),:)=d1d2s1s2(2+2*(m-1),:);
    sourcedelay(1+(num3-1)*4,:)=s1d2d1s2(1+2*(k-1),:);
    sourcedelay(2+(num3-1)*4,:)=d1d2s1s2(1+2*(m-1),:);
    sourcedelay(3+(num3-1)*4,:)=d1d2s1s2(2+2*(m-1),:);
    sourcedelay(4+(num3-1)*4,:)=s1d2d1s2(2+2*(k-1),:);
end
end
end

% If peaks are missing, calculate and then register them
s1d2d1s2left = s1d2d1s2-temp1;
left1 = abs(sum(s1d2d1s2left(:,2)));
if left1>0
templ1(:,1)=nonzeros(s1d2d1s2left(:,1));
templ1(:,2)=nonzeros(s1d2d1s2left(:,2));
templ1(:,3)=nonzeros(s1d2d1s2left(:,3));
templ1(:,4)=nonzeros(s1d2d1s2left(:,4));
end

did2s1s2left = did2s1s2-temp2;
left2 = abs(sum(did2s1s2left(:,2)));
if left2>0
temp22(:,1)=nonzeros(did2s1s2left(:,1));
temp22(:,2)=nonzeros(did2s1s2left(:,2));
temp22(:,3)=nonzeros(did2s1s2left(:,3));
temp22(:,4)=nonzeros(did2s1s2left(:,4));
end

if ( (exist('templ1')==1) & (exist('temp22')==1) )
sourcedelay(1,:) = templ1(1,2);
sourcedelay(2,:) = templ1(2,2)-templ1(1,4);
sourcedelay(3,:) = templ1(2,2)-templ1(2,4);
sourcedelay(4,:) = templ1(2,2);
sourcedelay(5,:) = temp22(2,2)-temp22(2,4);
sourcedelay(6,:) = temp22(1,2);
sourcedelay(7,:) = temp22(2,2);
sourcedelay(8,:) = temp22(2,2)-temp22(1,4);
elseif ( (exist('templ1')==1) & (exist('temp22')==0) )
    l11 = length(templ1(:,2))/2;
    for k=1:l11;
        num3 = num3+1;
        sourcedelay(1+4*(num3-1),1)=templ1(1+2*(k-1),2);
        sourcedelay(2+4*(num3-1),1)=templ1(2+2*(k-1),2)-...
            templ1(1+2*(k-1),4);
        sourcedelay(3+4*(num3-1),1)=templ1(2+2*(k-1),2)-...
            templ1(2+2*(k-1),4);
        sourcedelay(4+4*(num3-1),1)=templ1(2+2*(k-1),:);
    end
elseif ( (exist('templ1')==0) & (exist('temp22')==1) )
    l12 = length(temp22(:,2))/2;
    for k=1:l12;
        num3 = num3+1;
        sourcedelay(1+4*(num3-1),1)=temp22(2+2*(k-1),2)-...
            temp22(2+2*(k-1),4);
        sourcedelay(2+4*(num3-1),1)=temp22(1+2*(k-1),2);
        sourcedelay(3+4*(num3-1),1)=temp22(2+2*(k-1),2);
        sourcedelay(4+4*(num3-1),1)=temp22(2+2*(k-1),:)+...
            temp22(1+2*(k-1),:);
    end
elseif ( (exist('templ1')==0) & (exist('temp22')==0) )
    ls=length(sourcedelay)/4;
    if ls<2
        autotempl=auto1;
autotemp2=auto2;
for k=1:2;
autodiff1=autotemp1(k,2)-s1d2d1s2(1,4);
autodiff2=autotemp2(k,2)-s1d2d1s2(2,4);
if autodiff1==0
autotemp1(:,1)=zeros(1,3);
end
if autodiff2==0
autotemp2(:,1)=zeros(1,3);
end
end
s1d1=nonzeros(auto1(:,2)-autotemp1(:,2));
s2d2=nonzeros(auto2(:,2)-autotemp2(:,2));
um4=0;
for k=13:-1:2;
for m=k-1:-1:1;
differ1=abs(delaypeak(k,2)-delaypeak(m,2)-s1d1);
differ2=abs(delaypeak(k,2)-delaypeak(m,2)-s2d2);
if differ1<5
num4=num4+1;
sourcedelay(1+4*(num4-1),1)=(delaypeak(m,2)-s1d1)/8000;
sourcedelay(2+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcedelay(3+4*(num4-1),1)=(delaypeak(k,2))/8000;
sourcedelay(4+4*(num4-1),1)=(delaypeak(k,2)-s1d1)/8000;
sourcedelay(5+4*(num4-1),1)=(delaypeak(k,2))/8000;
sourcedelay(6+4*(num4-1),1)=(delaypeak(k,2)-s2d2)/8000;
sourcedelay(7+4*(num4-1),1)=(delaypeak(k,2))/8000;
sourcedelay(8+4*(num4-1),1)=(delaypeak(k,2)+s2d2)/8000;
save sourcedelay
end
end
end
end
elseif ( (exist('s1d2d1s2')==1) & (exist('s1d2d1s2')==0) )
ll1=length(s1d2d1s2(:,2))/2;
for k=1:ll1;
num3=num3+1;
sourcedelay(1+4*(num3-1),1)=s1d2d1s2(1+2*(k-1),2);
sourcedelay(2+4*(num3-1),1)=s1d2d1s2(2+2*(k-1),2);
sourcedelay(3+4*(num3-1),1)=s1d2d1s2(1+2*(k-1),4);
sourcedelay(4+4*(num3-1),1)=s1d2d1s2(2+2*(k-1),4);
end
if ll1<2
autotemp1=auto1;
autotemp2=auto2;
for k=1:2;
autodiff1=autotemp1(k,2)-s1d2d1s2(1,4);
autodiff2=autotemp2(k,2)-s1d2d1s2(2,4);
if autodiff1==0
autotemp1(:,1)=zeros(1,3);
end
if autodiff2==0
autotemp2(:,1)=zeros(1,3);
end
end
end
autotemp1(k,:)=zeros(1,3);
end
if autodiff2==0
autotemp2(k,:)=zeros(1,3);
end
end

s1d1=nonzeros(auto1(:,2)-autotempl(:,2));
s2d2=nonzeros(auto2(:,2)-autotemp2(:,2));
num4=0;
for k=13:-1:2;
for m=k-1:-1:1;
differ1=abs(delaypeak(k,2)-delaypeak(m,2)-s1d1);
differ2=abs(delaypeak(k,2)-delaypeak(m,2)-s2d2);
if differ1<5
num4=num4+1;
sourcedelay(1+4*(num4-1),1)=(delaypeak(m,2)-s2d2)/8000;
sourcedelay(2+4*(num4-1),1)=(delaypeak(k,2)-s2d2)/8000;
sourcedelay(3+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcedelay(4+4*(num4-1),1)=(delaypeak(k,2))/8000;
sourcedelay(6+4*(num4-1),1)=(delaypeak(k,2))/8000;
sourcedelay(7+4*(num4-1),1)=(delaypeak(m,2)+s2d2)/8000;
sourcedelay(8+4*(num4-1),1)=(delaypeak(k,2)+s2d2)/8000;
save sourcel sourcel
end
if differ2<5
fprintf('Possible TDOAs registrations for second source/n')
num4=num4+1;
sourcedelay(1+4*(num4-1),1)=(delaypeak(m,2)-s1d1)/8000;
sourcedelay(2+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcedelay(3+4*(num4-1),1)=(delaypeak(k,2)-s1d1)/8000;
sourcedelay(4+4*(num4-1),1)=(delaypeak(k,2))/8000;
sourcedelay(5+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcedelay(6+4*(num4-1),1)=(delaypeak(k,2))/8000;
sourcedelay(7+4*(num4-1),1)=(delaypeak(m,2)+s1d1)/8000;
sourcedelay(8+4*(num4-1),1)=(delaypeak(k,2)+s1d1)/8000;
save source2 source2
end
end
end
end
end
elseif( (exist('sid2dis2')==0) & (exist('did2sis2')==1) )
l22=length(did2sis2(:,2))/2;
for k=1:122;
num3=num3+1;
sourcedelay(1+4*(num3-1),1)=did2sis2(2+2*(k-1),2)-... did2sis2(2+2*(k-1),4);
sourcedelay(2+4*(num3-1),1)=did2sis2(1+2*(k-1),2);
sourcedelay(3+4*(num3-1),1)=did2sis2(2+2*(k-1),2);
sourcedelay(4+4*(num3-1),1)=did2sis2(2+2*(k-1),2)+... did2sis2(1+2*(k-1),4);
end
if l22<2
autotemp1=auto1;
autotemp2=auto2;
for k=1:2;
autodiff1=autotemp1(k,2)-did2sis2(1,4);
autodiff2=autotemp2(k,2)-did2sis2(2,4);
if autodiff1==0
autotemp1(k,:)=zeros(1,3);
end
if autodiff2==0
autotemp2(k,:)=zeros(1,3);
end

end

s1d=nonzeros(auto1(:,2)-autotemp1(:,2));
s2d=nonzeros(auto2(:,2)-autotemp2(:,2));
num4=0;
for k=13:-1:2;
for m=k-1:-1:1;

differ1=abs(delaypeak(k,2)-delaypeak(m,2)-s1d);
differ2=abs(delaypeak(k,2)-delaypeak(m,2)-s2d);
if differ1<5
num4=num4+1;
source1(1+4*(num4-1),1)=(delaypeak(m,2)-s2d)/8000;
source1(2+4*(num4-1),1)=(delaypeak(k,2)-s2d)/8000;
source1(3+4*(num4-1),1)=(delaypeak(k,2))/8000;
source1(4+4*(num4-1),1)=(delaypeak(k,2))/8000;
source1(5+4*(num4-1),1)=(delaypeak(m,2))/8000;
source1(6+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcel(7+4*(num4-1),1)=(delaypeak(k,2)+s2d)/8000;
sourcel(8+4*(num4-1),1)=(delaypeak(k,2)+s2d)/8000;
save sourcel sourcel
end
if differ2<5
num4=num4+1;
source2(1+4*(num4-1),1)=(delaypeak(m,2)-s1d)/8000;
source2(2+4*(num4-1),1)=(delaypeak(m,2))/8000;
source2(3+4*(num4-1),1)=(delaypeak(m,2))/8000;
source2(4+4*(num4-1),1)=(delaypeak(m,2))/8000;
source2(5+4*(num4-1),1)=(delaypeak(m,2))/8000;
source2(6+4*(num4-1),1)=(delaypeak(m,2)+s2d)/8000;
sourcel(7+4*(num4-1),1)=(delaypeak(k,2)+s2d)/8000;
sourcel(8+4*(num4-1),1)=(delaypeak(k,2)+s2d)/8000;
save source2 source2
end
end
end
elseif ( (exist('s1d2s2'))==0) & (exist('d1d2s1s2'))==0 )
num4=0;
for k=13:-1:2;
for m=k-1:-1:1;

differ1=abs(delaypeak(k,2)-delaypeak(m,2)-auto1(1,2));
differ2=abs(delaypeak(k,2)-delaypeak(m,2)-auto1(2,2));
differ3=abs(delaypeak(k,2)-delaypeak(m,2)-auto2(1,2));
differ4=abs(delaypeak(k,2)-delaypeak(m,2)-auto2(2,2));
if differ1<5
fprintf('Possible TDOAs registrations for a source/n')
num4=num4+1;
source1(1+4*(num4-1),1)=(delaypeak(m,2)-auto2(1,2))/8000;
source1(2+4*(num4-1),1)=(delaypeak(m,2)-auto2(1,2))/8000;
source1(3+4*(num4-1),1)=(delaypeak(m,2))/8000;
source1(4+4*(num4-1),1)=(delaypeak(m,2))/8000;
source1(5+4*(num4-1),1)=(delaypeak(m,2))/8000;
source1(6+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcel(7+4*(num4-1),1)=(delaypeak(m,2)+auto2(1,2))/8000;
sourcel(8+4*(num4-1),1)=(delaypeak(m,2)+auto2(1,2))/8000;
sourcel(9+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcel(10+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcel(11+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcel(12+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcel(13+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcel(14+4*(num4-1),1)=(delaypeak(m,2))/8000;
sourcel(15+4*(num4-1),1)=(delaypeak(m,2)/8000;
sourcel(16+4*(num4-1),1)=(delaypeak(m,2)/8000;
save sourcel sourcel
end
end
end
end

save source1 source1
end
if differ2<5
fprintf('Possible TDOAs registrations for a source/s')
num4=num4+1;
source2(1+4*(num4-1),1)=(delaypeak(m,2)-auto2(2,2))/8000;
source2(2+4*(num4-1),1)=(delaypeak(k,2)-auto2(2,2))/8000;
source2(3+4*(num4-1),1)=(delaypeak(m,2))/8000;
source2(4+4*(num4-1),1)=(delaypeak(k,2))/8000;
source2(5+4*(num4-1),1)=(delaypeak(m,2)+auto2(2,2))/8000;
source2(6+4*(num4-1),1)=(delaypeak(k,2)+auto2(2,2))/8000;
source2(7+4*(num4-1),1)=(delaypeak(m,2)-auto2(2,2))/8000;
source2(8+4*(num4-1),1)=(delaypeak(k,2)-auto2(2,2))/8000;
save source2 source2
end
if differ3<5
fprintf('Possible TDOAs registrations for a source/s')
num4=num4+1;
source3(1+4*(num4-1),1)=(delaypeak(m,2)-auto(1,2))/8000;
source3(2+4*(num4-1),1)=(delaypeak(m,2))/8000;
source3(3+4*(num4-1),1)=(delaypeak(k,2)-auto(1,2))/8000;
source3(4+4*(num4-1),1)=(delaypeak(k,2))/8000;
source3(5+4*(num4-1),1)=(delaypeak(m,2))/8000;
source3(6+4*(num4-1),1)=(delaypeak(m,2)-auto(1,2))/8000;
source3(7+4*(num4-1),1)=(delaypeak(k,2))/8000;
save source3 source3
end
if differ4<5
fprintf('Possible TDOAs registrations for a source/s')
num4=num4+1;
save source4 source4
end
end
3. File name: check2.m

% CHECK2.M : this subprogram checks the delay pairs s1d2, s1s2 and % d1d2, d1s2 for the case when only all autocorrelation delays at the % sensor 1 are present and one autocorrelation delay at sensor 2 % taken into account the possibility of missing peaks.

suma11=autol(1,2)+autol(1,2);
suma12=autol(2,2)+autol(2,2);

for k=13:-1:2;
for m=k-1:-1:1;

diff1=abs(delaypeak(k,2)-delaypeak(m,2)-sumall);
diff2=abs(delaypeak(k,2)-delaypeak(m,2)-sumal2);
if diff1<5
    count1=count1+1;
s1d2s1d2(1+(count1-1)*2:1:3)=delaypeak(m,);
s1d2s1d2(1+(count1-1)*2:1:3)=delaypeak(k,);
s1d2s1d2(2+(count1-1)*2:1:3)=autol(1,2);
s1d2s1d2(2+(count1-1)*2:1:3)=autol(1,2);
elseif diff2<5
    count1=count1+1;
s1d2s1d2(1+(count1-1)*2:1:3)=delaypeak(m,);
s1d2s1d2(1+(count1-1)*2:1:3)=delaypeak(k,);
s1d2s1d2(2+(count1-1)*2:1:3)=autol(2,2);
s1d2s1d2(2+(count1-1)*2:1:3)=autol(2,2);
end

subal1=-autol(1,2)+autol(1,2);
subal2=-autol(2,2)+autol(2,2);

diff3=abs(delaypeak(k,2)-delaypeak(m,2)-suba11);
diff4=abs(delaypeak(k,2)-delaypeak(m,2)-suba12);
if diff3<5
    count2=count2+1;
d1d2s1d2(1+(count2-1)*2:1:3)=delaypeak(m,);
d1d2s1d2(1+(count2-1)*2:1:3)=delaypeak(k,);
d2s1s2(2+(count2-1)*2:1:3)=autol(1,2);
d2s1s2(2+(count2-1)*2:1:3)=autol(1,2);
elseif diff4<5
    count2=count2+1;
d1d2s1d2(1+(count2-1)*2:1:3)=delaypeak(m,);
d1d2s1d2(1+(count2-1)*2:1:3)=delaypeak(k,);
d2s1s2(2+(count2-1)*2:1:3)=autol(2,2);
d2s1s2(2+(count2-1)*2:1:3)=autol(2,2);
end

end

% classifying the TDOA estimates to the corresponding source based % on available information and giving possible results

if ( (exist('s1d2s1d2')==1) & (exist('d1d2s1s2')==1) )
    num1=length(s1d2s1d2(:,2))/2;
    num2=length(d1d2s1s2(:,2))/2;
    temp1=zeros(size(s1d2s1d2));
    temp2=zeros(size(d1d2s1s2));
    num3=0;
    for k=1:num1;
        for m=1:num2;
sum1=abs( sld2d1s2(1+2*(k-1),2)+s1d2d1s2(2+2*(k-1),2)...
-d1d2s1s2(1+2*(m-1),2)-d1d2s1s2(2+2*(m-1),2) );

if sum1<5
num3=num3+1;
source1(1+(num3-1)*4,1)=s1d2d1s2(1+2*(k-1),2);
sourcel(2+(num3-1)*4,1)=s1d2d1s2(1+2*(k-1),2)+sid2d1s2(1+2*(k-1),4);
sourcel(3+(num3-1)*4,1)=sid2d1s2(1+2*(k-1),2);
sourcel(4+(num3-1)*4,1)=s1d2d1s2(2+2*(k-1),2);
save sourcel sourcel1 end end

end

for m=1:2;
autodiff=abs(auto1(m,2)-sid2d1s2(1,4));
if autodiff>0
sldl=auto1(m,2);
end end

end

elseif ( (exist('sid2d1s2')==1) & (exist('d1d2s1s2')==0) )
num1=length(sid2d1s2(:,2))/2;
for k=1:num1;
source1(1+(num3-1)*4,1)=sid2d1s2(1+2*(k-1),2);
sourcel(2+(num3-1)*4,1)=sid2d1s2(1+2*(k-1),2)+sid2d1s2(1+2*(k-1),4);
sourcel(3+(num3-1)*4,1)=sid2d1s2(1+2*(k-1),2);
sourcel(4+(num3-1)*4,1)=sid2d1s2(2+2*(k-1),2);
save sourcel sourcel1 end end

end

for m=1:2;
autodiff=abs(auto1(m,2)-sid2d1s2(1,4));
if autodiff>0
sldl=auto1(m,2);
end end

end

elseif ( (exist('sid2d1s2')==0) & (exist('d1d2s1s2')==1) )
num2=length(d1d2s1s2(:,2))/2;
for k=1:num1;
source1(1+(num3-1)*4,1)=d1d2s1s2(1+2*(k-1),2);
sourcel(2+(num3-1)*4,1)=d1d2s1s2(1+2*(k-1),2);
sourcel(3+(num3-1)*4,1)=d1d2s1s2(2+2*(k-1),2);
sourcel(4+(num3-1)*4,1)=d1d2s1s2(2+2*(k-1),2)+d1d2s1s2(1+2*(k-1),4);
save sourcel sourcel1 end end

end

for m=1:2;
autodiff=abs(auto1(m,2)-d1d2s1s2(1,4));
if autodiff>0
sldl=auto1(m,2);
end end

end

num4=0;
for k=13:-1:2;
for m=k-1:-1:1;
differ=abs(delaypeak(k,2)-delaypeak(m,2)-sldl);
end
end

endif
if differ1<5
num4=num4+1;
comb(1+2*(num4-1),1)=delaypeak(k,2);
comb(2+2*(num4-1),1)=delaypeak(m,2);
end
end
end
if (exist('comb')==1)
14=length(tdoapair)/2;
num5=0;
if 14>1
for k=1:14-1;
for m=1:14-k;
differ2=abs(comb(k,2)+comb(2+2*m,2)-... 
com2(k,2)+comb(k+1,2));
if differ2<5
num5=num5+1;
sourc2(1*4*(num5-1))=comb(2+2*m);
sourc2(2+4*(num5-1))=comb(1+2*m);
sourc2(3+4*(num5-1))=comb(k+1,2);
sourc2(4+4*(num5-1))=comb(k,2);
save sourc2 sourc2
end
end
end
else
fprintf('Classification for the second source is not possible/n')
end
elseif (exist('comb')==0)
fprintf('Classification for the second source is not possible/n')
end
elseif ( (exist('s1d2d1s2')=='0') & (exist('did2s1s2')=='0') )
for k=13:-1:2;
for m=k-1:-1:1;
diff1=abs(delaypeak(k,2)-delaypeak(m,2)-auto2(1,2));
diff2=abs(delaypeak(k,2)-delaypeak(m,2)-auto2(2,2));
diff3=abs(delaypeak(k,2)-delaypeak(m,2)-auto2(1,2));
if diff1<5
count1=count1+1;
sourc1(1+4*(count1-1))=delaypeak(m,2);
sourc1(2+4*(count1-1))=delaypeak(k,2);
sourc1(3+4*(count1-1))=delaypeak(m,2)+auto2(1,2);
sourc1(4+4*(count1-1))=delaypeak(k,2)+auto2(1,2);
sourc1(5+4*(count1-1))=delaypeak(m,2)-auto2(1,2);
sourc1(6+4*(count1-1))=delaypeak(k,2)-auto2(1,2);
sourc1(7+4*(count1-1))=delaypeak(m,2);
sourc1(8+4*(count1-1))=delaypeak(k,2);
sourc1=sourc1';
save sourc1 sourc1
end
if diff2<5
count2=count2+1;
sourc2(1+4*(count2-1))=delaypeak(m,2);
sourc2(2+4*(count2-1))=delaypeak(k,2);
sourc2(3+4*(count2-1))=delaypeak(m,2)+auto2(1,2);
sourc2(4+4*(count2-1))=delaypeak(k,2)+auto2(1,2);
sourc2(5+4*(count2-1))=delaypeak(m,2)-auto2(1,2);
sourc2(6+4*(count2-1))=delaypeak(k,2)-auto2(1,2);
source2(7+4*(count1-1))=delaypeak(m,2);
source2(8+4*(count1-1))=delaypeak(k,2);
source2=source2';
save source2 source2
end
if diff3<5
count2=count2+1;
source3(1+4*(count1-1))=delaypeak(m,2);
source3(2+4*(count1-1))=delaypeak(k,2)+auto1(1,2);
source3(3+4*(count1-1))=delaypeak(k,2);
source3(4+4*(count1-1))=delaypeak(k,2)+auto1(1,2);
source3(5+4*(count1-1))=delaypeak(k,2)-auto2(1,2);
source3(6+4*(count1-1))=delaypeak(m,2);
source3(7+4*(count1-1))=delaypeak(k,2)-auto2(1,2);
source3(8+4*(count1-1))=delaypeak(k,2);
source3=source3';
save source3 source3
end
end
end
% end of check2.m

4. File name: checksid1.m
% CHECKSID1: check pairs of TDOAs which meet the condition
% sidl=s1d1-d1d2=s1s2-d1s2

num4=0;
for k=13:-1:2;
for m=k-1:-1:1;
differ1=abs(delaypeak(k,2)-delaypeak(m,2)-s1d1);
if differ1<5
num4=num4+1;
comb(1+2*(num4-1),1)=delaypeak(k,2);
comb(2+2*(num4-1),1)=delaypeak(m,2);
end
end
end
if (exist('comb')==1)
14=length(tdoapair)/2;
um5=0;
if 14>1
for k=1:14-1;
for m=1:14-k;
differ2=abs(comb(k,2)+comb(2+2*m,2)-comb(1+2*m,2)-comb(k+1,2));
if differ2<5
num5=num5+1;
source2(1+4*(num5-1))=comb(2+2*m);
source2(2+4*(num5-1))=comb(1+2*m);
source2(3+4*(num5-1))=comb(k+1,2);
source2(4+4*(num5-1))=comb(k,2);
save source2 source2
end
end
end
else
fprintf('Classification for the second source is not possible\n')
end

4. File name: checksid1.m
% CHECKSID1: check pairs of TDOAs which meet the condition
% sidl=s1d1-d1d2=s1s2-d1s2

num4=0;
for k=13:-1:2;
for m=k-1:-1:1;
differ1=abs(delaypeak(k,2)-delaypeak(m,2)-s1d1);
if differ1<5
num4=num4+1;
comb(1+2*(num4-1),1)=delaypeak(k,2);
comb(2+2*(num4-1),1)=delaypeak(m,2);
end
end
end
if (exist('comb')==1)
14=length(tdoapair)/2;
um5=0;
if 14>1
for k=1:14-1;
for m=1:14-k;
differ2=abs(comb(k,2)+comb(2+2*m,2)-comb(1+2*m,2)-comb(k+1,2));
if differ2<5
num5=num5+1;
source2(1+4*(num5-1))=comb(2+2*m);
source2(2+4*(num5-1))=comb(1+2*m);
source2(3+4*(num5-1))=comb(k+1,2);
source2(4+4*(num5-1))=comb(k,2);
save source2 source2
end
end
end
else
fprintf('Classification for the second source is not possible\n')
end
elseif (exist('comb')==0)
fprintf('Classification for the second source is not possible\n')
end
% end of checksid1.m

5. File name: check21.m
% CHECK21.M : this subprogram checks the delay pairs s1d2, s1s2 and
% d1d2, d1s2 for the case when only all autocorrelation delays at
% sensor 1 are present and takes into account the possibility of
% missing peaks
for k=13:-1:2;
for m=k-1:-1:1;

diff1=abs(delaypeak(k,2)-delaypeak(m,2)-autol(1,2));
diff2=abs(delaypeak(k,2)-delaypeak(m,2)-autol(2,2));

if diff1<5
    count1=count1+1;
    comb1(1+(count1-1)*2,1:3)=delaypeak(k,:);
    comb1(1+(count1-1)*2,4)=autol(1,2);
    comb1(2+(count1-1)*2,1:3)=delaypeak(m,:);
    comb1(2+(count1-1)*2,4)=autol(1,2);
end
if diff2<5
    count2=count2+1;
    comb2(1+(count2-1)*2,1:3)=delaypeak(k,:);
    comb2(1+(count2-1)*2,4)=autol(2,2);
    comb2(2+(count2-1)*2,1:3)=delaypeak(m,:);
    comb2(2+(count2-1)*2,4)=autol(2,2);
end
end
end

n1=length(comb1)/2;
count3=0;
for k=1:n1-1;
m=1:n1-k;

diff3=abs( comb1(k,2)+comb1(2+2*m,2)-comb1(1+2*m,2)-comb1(k+1,2) );
if diff3<5
    count3=count3+1;
    source1(1+4*(count3-1))=comb1(2+2*m,2);
    source1(2+4*(count3-1))=comb1(1+2*m,2);
    source1(3+4*(count3-1))=comb1(k+1,2);
    source1(4+4*(count3-1))=comb1(k,2);
end
end

e1=length(comb2)/2;
count4=0;
for k=1:e1-1;
m=1:e1-k;

diff4=abs( comb2(k,2)+comb2(2+2*m,2)-comb2(1+2*m,2)-comb2(k+1,2) );
if diff4<5
    count4=count4+1;
    source2(1+4*(count4-1))=comb2(2+2*m,2);
    source2(2+4*(count4-1))=comb2(1+2*m,2);
    source2(3+4*(count4-1))=comb2(k+1,2);
    source2(4+4*(count4-1))=comb2(k,2);
end
end
if (exist('source1')==1) & (exist('source2')==1) 
    sourcedelay(:,1)=[source1,source2];
elseif (exist('source1')==1) 
    source1=source1'; 
    save source1 source1 
elseif (exist('source2')==1) 
    source2=source2'; 
    save source2 source2 
end 
% end of check21.m

6. File name: check3.m
% CHECK3.M : this subprogram checks the delay pairs s1d2,s1s2 and 
% d1d2,d1s2 for the case when only all autocorrelation delays at sensor 1 
% are present and takes into account the possibility of missing peaks 

for k=13:-1:2; 
    for m=k-1:-1:1; 
        diffl=abs(delaypeak(k,2)-delaypeak(m,2)-auto2(1,2)); 
        diff2=abs(delaypeak(k,2)-delaypeak(m,2)-auto2(2,2)); 
        if diffl<5 
            count1=count1+1; 
            comb1(1+(count1-1)*2,1:3)=delaypeak(k,:); 
            comb1(1+(count1-1)*2,4)=auto2(1,2); 
            comb1(2+(count1-1)*2,1:3)=delaypeak(m,:); 
            comb1(2+(count1-1)*2,4)=auto2(1,2); 
        end 
        if diff2<5 
            count2=count2+1; 
            comb2(1+(count2-1)*2,1:3)=delaypeak(k,:); 
            comb2(1+(count2-1)*2,4)=auto2(2,2); 
            comb2(2+(count2-1)*2,1:3)=delaypeak(m,:); 
            comb2(2+(count2-1)*2,4)=auto2(2,2); 
        end 
    end 
end

l11=length(comb1)/2; 
count3=0; 
for k=1:l11-1; 
    m=1:l11-k; 
    diff3=abs( comb1(k,2)+comb1(2+2*m,2)-comb1(k+1,2)-comb1(1+2*m,2 ) ); 
    if diff3<5 
        count3=count3+1; 
        source1((1+4*(count3-1)))=comb1(2+2*m,2); 
        source1((2+4*(count3-1)))=comb1(k+1,2); 
        source1((3+4*(count3-1)))=comb1(1+2*m,2); 
        source1((4+4*(count3-1)))=comb1(k,2); 
    end 
end 

l12=length(comb2)/2; 
count4=0; 
for k=1:l12-1; 
    m=1:l12-k; 
    diff4=abs( comb2(k,2)+comb2(2+2*m,2)-comb2(k+1,2)-comb2(1+2*m,2 ) ); 
    if diff4<5 
        count4=count4+1; 
        source2((1+4*(count4-1)))=comb2(2+2*m,2); 
    end 
end
source2(2+4*(count4-1))=comb2(k+1,2);
sorce2(3+4*(count4-1))=comb2(1+2*m,2);
sorce2(4+4*(count4-1))=comb2(k,2);
end
end
end

if (exist('sourcel')==1) & (exist('source2')==1)
sourcedelay(:,1)=[sourcel,source2]';
elseif (exist('source1')==1)
sourcel=source1';
save sourcel sourcel
elseif (exist('source2')==1)
sourcel=source2';
save source2 source2
end
% end of check3.m

7. File name: check4.m
% CHECK4.M: This subprogram checks the TDOA estimates in the case only % one delay was detected in each autocorrelation and takes into account % the possibility of missing peaks

% Check if only one source is present
suma=auto1(2)+auto2(2);
suba=auto1(2)+auto2(2);
for k=13:-1:2;
    for m=k-1:-1:1;
        diff1=abs(delaypeak(k,2)-delaypeak(m,2)-suma);
        if diff1<5
            count1=count1+1;
sid2dis2(1+(count1-1)*2,1:3)=delaypeak(m,:);
sid2dis2(1+(count1-1)*2,4)=auto1(1,2);
sid2dis2(2+(count1-1)*2,1:3)=delaypeak(k,:);
sid2dis2(2+(count1-1)*2,4)=auto2(1,2);
        end
        diff2=abs(delaypeak(k,2)-delaypeak(m,2)-suba);
        if diff2<5
            count2=count2+1;
did2dis2(1+(count2-1)*2,1:3)=delaypeak(m,:);
did2dis2(1+(count2-1)*2,4)=auto1(1,2);
did2dis2(2+(count2-1)*2,1:3)=delaypeak(k,:);
did2dis2(2+(count2-1)*2,4)=auto2(1,2);
        end
    end
end
if ( (exist('sid2dis2')==1) & (exist('did2dis2')==1) )
    num1=length(sid2dis2(:,2))/2;
    num2=length(did2dis2(:,2))/2;
    num3=0;
    for k=1:num1;
        for m=1:num2;
            sum1=abs( sid2dis2(1+2*(k-1),2)+sid2dis2(2+2*(k-1),2)... -did2dis2(1+2*(m-1),2)-did2dis2(2+2*(m-1),2) );
            if sum1<5
                num3=num3+1;
sourcedelay(1+4*(num3-1),1)=sid2dis2(1+2*(k-1),2);
sourcedelay(2+4*(num3-1),1)=did2dis2(1+2*(m-1),2);
sourcedelay(3+4*(num3-1),1)=sid2dis2(2+2*(k-1),2);
sourcedelay(4+4*(num3-1),1)=did2dis2(2+2*(m-1),2);
        end
    end
end
sourcedelay(4*4*(num3-1),1)=std2dis2(2+2*(k-1),2);
end
end
end
else
for k=13:-1:2;
for m=k-1:-1:1;

diff1=abs(delaypeak(k,2)-delaypeak(m,2)-autol(2));
diff2=abs(delaypeak(k,2)-delaypeak(m,2)-auto2(2));

if diff1<5
count1=count1+1;
combi(1+(count1-1)*2,1:3)=delaypeak(k,:);
combi(1+(count1-1)*2,4)=autol(2);
combi(2+(count1-1)*2,1:3)=delaypeak(m,:);
combi(2+(count1-1)*2,4)=autol(2);
end

if diff2<5
count2=count2+1;
comb2(1+(count2-1)*2,1:3)=delaypeak(k,:);
comb2(1+(count2-1)*2,4)=auto2(2);
comb2(2+(count2-1)*2,1:3)=delaypeak(m,:);
comb2(2+(count2-1)*2,4)=auto2(2);
end
end
end

k=1:111-1;
for m=1:111-k;

diff3=abs( combi(k,2)+comb2(2+2*m,2)-comb2(1+2*m,2)-comb1(k+1,2) );
if diff3<5
count3=count3+1;
sourcel(1+4*(count3-1))=comb1(2+2*m,2);
sourcel(2+4*(count3-1))=comb2(1+2*m,2);
sourcel(3+4*(count3-1))=comb1(k+1,2);
sourcel(4+4*(count3-1))=combi(k,2);
end
end
end

k=1:122-1;
for m=1:122-k;

diff4=abs( comb2(k,2)+comb2(2+2*m,2)-comb1(1+2*m,2)-comb2(k+1,2) );
if diff4<5
count4=count4+1;
source2(1+4*(count4-1))=comb2(2+2*m,2);
source2(2+4*(count4-1))=comb2(k+1,2);
source2(3+4*(count4-1))=comb1(k,2);
source2(4+4*(count4-1))=combi(k,2);
end
end
end

if (exist('sourcel')==1) & (exist('source2')==1)
sourcedelay(:,1)=[sourcel,source2]';
else if (exist('sourcel')==1)
sourcel=sourcel';
save sourcel sourcel
else if (exist('source2')==1)
source2=source2';
save source2 source2
end
end
save source2 source2
end

end
% end of check4.m

8. File name: check5.m
% CHECK5.M: This subprogram classifies the TDOA estimates in the case
% only one delay was detected in auto1.

for k=13:-1:2;
for m=k-1:-1:1;
diffl=abs(delaypeak(k,2)-delaypeak(m,2)-auto1(2));
if diffl<5
    countl=countl+1;
    comb1(1+(countl-1)*2,1:3)=delaypeak(k,:);
    comb1(1+(countl-1)*2,4)=auto1(2);
    comb1(2+(countl-1)*2,1:3)=delaypeak(m,:);
    comb1(2+(countl-1)*2,4)=auto1(2);
end
end
end

111=length(comb1)/2;
count3=0;
for k=1:111-1;
m=1:111-k;
diff2=abs(comb1(k,2)+comb1(2+2*m,2)-comb1(1+2*m,2)-comb1(k+1,2));
if diff2<5
    count3=count3+i;
    source1(1+4*(count3-1))=comb1(2+2*m,2);
    sourcel(2+4*(count3-1))=comb1(1+2*m,2);
    source1(3+4*(count3-1))=comb1(k+1,2);
    sourcel(4+4*(count3-1))=comb1(k,2);
end
end
end

if (exist('source1')==1)
source1=sourcel';
save sourcel sourcel
end
% end of check5.m

9. File name: check6.m
% CHECK6.M: This subprogram classifies the TDOA estimates in the case
% only one delay was detected in auto2 with the possibility of missing
% peaks taken into account

for k=13:-1:2;
for m=k-1:-1:1;
diff1=abs(delaypeak(k,2)-delaypeak(m,2)-auto2(2));
if diff1<5
    count2=count2+1;
    comb2(1+(count2-1)*2,1:3)=delaypeak(k,:);
    comb2(1+(count2-1)*2,4)=auto2(2);
    comb2(2+(count2-1)*2,1:3)=delaypeak(m,:);
    comb2(2+(count2-1)*2,4)=auto2(2);
end
end
end

if (exist('source1')==1)
source1=sourcel';
save sourcel sourcel
end
% end of check6.m
\texttt{end}
\texttt{end}
\texttt{end}

\texttt{122=\texttt{length(comb2)/2};}
\texttt{count4=0;}
\texttt{for \texttt{k=}1:122-1;}
\texttt{m=1:122-k;}

\texttt{diff4=abs( comb2(k,2)+comb2(2+2*m,2)-comb2(1+2*m,2)-comb2(k+1,2) );}
\texttt{if \texttt{diff4<5}}
\texttt{count4=\texttt{count4+1};}
\texttt{source1(1+4*(count4-1))=comb2(2+2*m,2);}
\texttt{source1(2+4*(count4-1))=comb2(k+1,2);}
\texttt{source1(3+4*(count4-1))=comb2(1+2*m,2);}
\texttt{source1(4+4*(count4-1))=comb2(k,2);}
\texttt{end}
\texttt{end}
\texttt{end}

\texttt{if (exist('source1')==1)}
\texttt{source1=source1';}
\texttt{save source1 source1}
\texttt{end}
\texttt{\% end of check6.m}