

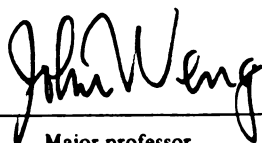


132
606
THS



This is to certify that the
thesis entitled
**Learning-Based Three Dimensional Sound Localization
Using a Compact Non-Coplanar Array of Microphones**
presented by
Kamen Yankov Guentchev

has been accepted towards fulfillment
of the requirements for
Master of Science degree in **Computer Science**


Major professor

Date 12 / 08 / 1997

LIBRARY
Michigan State
University

PLACE IN RETURN BOX
to remove this checkout from your record.
TO AVOID FINES return on or before date due.

DATE DUE	DATE DUE	DATE DUE
<hr/>	<hr/>	<hr/>
<hr/>	<hr/>	<hr/>
<hr/>	<hr/>	<hr/>
<hr/>	<hr/>	<hr/>
<hr/>	<hr/>	<hr/>

LEARNING-BASED THREE DIMENSIONAL SOUND LOCALIZATION
USING A COMPACT NON-COPLANAR ARRAY OF MICROPHONES

By

Kamen Yankov Guentchev

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Computer Science Department

1997

ABSTRACT

LEARNING-BASED THREE DIMENSIONAL SOUND LOCALIZATION USING A COMPACT NON-COPLANAR ARRAY OF MICROPHONES

By

Kamen Yankov Guentchev

One of the various human sensory capabilities is to identify the direction of perceived sounds. The goal of this work is to study sound source localization in three dimensions using some of the most important cues the human uses. Having robotics as a major application, the approach involves a compact sensor structure that can be placed on a mobile platform. The objective is to estimate the relative sound source position in three dimensional space without imposing excessive restrictions on its spatio-temporal characteristics and the environment structure. Two types of features are considered, interaural time and level differences. Their relative effectiveness for localization is studied, as well as a practical way of using these complementary parameters. A two-stage procedure was used. In the training stage, sound samples are produced from points with known coordinates and then are stored. In the recognition stage, unknown sounds are processed by the trained system to estimate the 3D location of the sound source. Results from the experiments showed under $\pm 3^\circ$ in average angular error and less than $\pm 20\%$ in average radial distance error.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. John Weng for the productive guidance throughout my thesis research. Special thanks go to Dr. George Stockman for his support in my studies and scientific work in general, and for believing that an astronomer can be a good computer scientist, too. I would also like to thank the other members of my committee, Dr. Moon-Jung Chung and Dr. Sridhar Mahadevan, for the useful comments and suggestions. I am very grateful to Shaoyun Chen for making SHOSLIF code available for use by this project and for having helped in many other ways. I would also like to thank all students in Pattern Recognition and Image Processing laboratory for assisting me in my experimental work and in the preparation of this thesis. Last but certainly not least, I would like to thank my dearest fiancée, Galia, who helped me in so many ways. I could not have completed this work without her invaluable help. She not only spent many sleepless nights helping me take samples from the experimental setup, preparing tables and graphics and proofing copies of the thesis but most importantly she always supported me morally and never let me stop believing I could achieve this goal.

TABLE OF CONTENTS

LIST OF TABLES	v
LIST OF FIGURES	vi
1 Introduction	1
1.1 The problem of sound localization by machine	2
1.2 Review of related experimental work	5
2 Theoretical problem and sensor structure	7
2.1 Parameters	8
2.2 Number and placement of detectors	9
2.3 Geometric representation	10
2.3.1 From Intensity Ratios (IR)	10
2.3.2 From Phase Differences (PD)	14
3 Methodology	16
3.1 Feature Extraction	16
3.2 Source localization	19
4 Experimental setup and results	21
4.1 Experiment and results	21
4.1.1 Hardware	22
4.1.2 Experiment	24
4.1.3 Results	26
4.2 Program performance and details	29
4.3 Implementation restrictions	30
5 Conclusions	38
5.1 Discussion	38
5.2 Future research	39
APPENDICES	41
A Class “IADiff”: Signal Preprocessing	41
B Class “LinSrch”: Linear Search	48

LIST OF TABLES

4.1	Comparative timings of various routines	28
4.2	Average error in azimuth for different values of <i>Scaling on ILD</i> (x) and <i>weighting coefficient</i> (y) as plotted in Figure. 4.3 [degrees]	32
4.3	Standard deviation of error in azimuth, for different values of <i>Scaling on ILD</i> (x) and <i>weighting coefficient</i> (y) as plotted in Figure. 4.4 [degrees]	33
4.4	Average error in elevation for different values of <i>Scaling on ILD</i> (x) and <i>weighting coefficient</i> (y) as plotted in Figure. 4.5 [degrees]	34
4.5	Standard deviation of error in elevation, for different values of <i>Scaling on ILD</i> (x) and <i>weighting coefficient</i> (y) as plotted in Figure. 4.6 [degrees]	34
4.6	Average error in distance for different values of <i>Scaling on ILD</i> (x) and <i>weighting coefficient</i> (y) as plotted in Figure. 4.7 [%]	35
4.7	Standard deviation of error in distance, for different values of <i>Scaling on ILD</i> (x) and <i>weighting coefficient</i> (y) as plotted in Figure. 4.8 [%]	35
4.8	Error values for <i>Scaling on ILD</i> = 11 and <i>weighting coefficient</i> = 1	36

LIST OF FIGURES

2.1	Two detectors and a sound source	8
2.2	Microphone placement on the array	10
2.3	Distances from source to four detectors	11
3.1	Preselection according to signal variance	17
3.2	Typical cross-correlation curve with good sharpness	18
4.1	Training the system	23
4.2	Experimental setup	24
4.3	Distribution of error values for Azimuth	26
4.4	Distribution of standard deviation for Azimuth	27
4.5	Distribution of error values for Elevation	28
4.6	Distribution of standard deviation for Elevation	29
4.7	Distribution of error values for Distance	30
4.8	Distribution of standard deviation for Distance	31
4.9	Screen with signals and data dialog	37

Chapter 1

Introduction

A sound produced by a point-source generates acoustic waves with spherical symmetry, assuming uniform density of the surrounding air and absence of obstacles or other sounds. It is known that the location of the source can be established by detecting the front of the propagating wave and computing the center of the sphere [9][10]. Unfortunately acoustical waves are not clearly distinguishable objects and such a task is not trivial in real environments even if real-life sources could be approximated by points [18]. Numerous studies have attempted to determine the mechanisms used by humans to achieve dimensional hearing [10][13][14]. Most phenomena have been reasonably explained in principle, although many aspects of human dimensional hearing need further study. It is known that two of the most important cues used by humans are the interaural differences: in time and level (ITD, ILD) [18][24][25]. Other cues relate to the spectral variations caused by diffractions at the head and pinnae [3]. For sounds with longer duration, cognitive processes start playing an important role, including dynamic head adjustments, high-level reasoning, etc. [25].

1.1 The problem of sound localization by machine

Sound localization can be used in many different applications: robot hearing, human-machine interfaces, monitoring devices, handicappers' aids, etc, where other means fail for different reasons. The obvious importance of building sound localization devices has prompted numerous efforts in the research community and a variety of techniques has been developed. Driven by concrete application needs, sensor setups of different implementations have seldom attempted to follow the human model. The number, size and placement of the sensors in such devices follow the specific needs of the task and are optimized for accuracy, stability, ease of use, etc. For example, a number of microphone subarrays have been placed on the walls with a goal to pick up the location of a speaker in a room [5][6][7][20]. In other studies a human model has been followed to some degree resulting in constraints in applicability and limited accuracy [19]. A significant amount of work has been devoted to devices with a limited functionality (e.g. constrained to localization in a single half-plane while still using large sensor structures) [8][20] or the help of a non-acoustical modality has been used (e.g. vision)[8].

In contrast to large, fixed sensor arrays for special situations and environments, this work concentrates on a compact, mobile sensor array that is suited for a mobile robot to localize 3D sound sources with moderate accuracy. It should have the ability to be positioned arbitrarily in space while being capable of identifying the relative position of an arbitrarily located sound source. It is necessary to point out that the human three dimensional sound localization capabilities, while amazingly accurate in some instances often have very serious limitations. The precision depends on various characteristics of the perceived sound: spectral contents, envelope variability as a function of time, volume level, reverberation and echo, etc. It can be disappointingly

low and in some instances totally inconclusive [13]. Sometimes it can be convincingly wrong (e.g. Franssen effect) [14]. One major difference between human and engineering setup is the number of sensors available.

Most authors distinguish a single parameter as the most significant factor for dimensional sound localization. It is the interaural time difference (ITD) of the sound as perceived by two sensors. Numerous studies report the ITD as the main cue in human dimensional hearing [24]. The clear geometrical representation of the problem makes it the favorite feature to be used when approaching such a task by a machine setup [4][5][7][8][12][16][17][20]. However, if we return to the physical aspect of the problem, it is clear that even three sensors (non-collinear) are not sufficient by themselves to establish unambiguously the three dimensional location of the sound source: obviously there are two symmetrical solutions on each side of the plane, containing the sensors. It is then reasonable to increase the number to four microphones, to localize arbitrarily placed sound sources.

Another cue known to have notable importance in human dimensional hearing is the interaural level differences (ILD). Surprisingly ILD have seldom been used in actual system implementations because they are believed to have unfavorable frequency dependence and unreliability [18][19]. Another reason is the lack of an explicit and stable relationship between ILD and source location which will allow for a simple algorithmic solution to be derived [18]. The learning approach used in this study does not have such limitations and it benefits from the added cues.

Finally the processing of the extracted features is one of the dominating factors for the success of a localization procedure. Most works determine the ITD and then use either an iterative search algorithm to minimize a certain objective function [15][19][20], or an approximation model for which a closed-form solution can be derived [5][6]. The former is relatively slow and thus, it may not reach real time speed.

The latter introduces model errors and cannot use more features for better accuracy.

To use both interaural time differences (ITD) and interaural level differences (ILD) while effectively dealing with the complex nonlinear relationships among these feature measurements and the solution, this work employs a learning based approach. It consists of a training phase and a performance phase. In the training phase, sounds from known 3D positions are generated for training the system, which builds a fast retrieval tree. In the performance phase, the system approximates the solution by retrieving the top match cases from the retrieval tree. This flexible framework allows for the use of more than one type of feature, and to deal with the 3D localization problem without imposing unrealistic assumptions about the environment, despite the compactness of the sensor structure. To the best of my knowledge, this work is the first to use a compact non-coplanar sensor array for full 3D sound localization.

In order to correctly evaluate the performance of the system, initially a linear search algorithm was used when searching for the nearest neighbors in the 12-dimensional input space. The obtained results were also used to evaluate the correctness and the performance of the previously developed and tested with other implementations SHOSLIF procedure. It achieves a high speed of retrieval due to its logarithmic time complexity $O(\log(n))$, where n is the number of cases learned and stored as necessary [22][23]. It was found that the results produced by SHOSLIF had identical precision with that of the linear search, while its performance time was nearly 5 times faster, even with the relatively low dimensionality of the employed search space.

1.2 Review of related experimental work

Extensive work on sound localization by microphone arrays has been performed by Michael Brandstein, in association with other authors at Brown University. A significant effort in the theory of speech based sound localization is incorporated in a series of publications [4][5][6][7]. Novel methods for estimating ITD and using it for the localization are presented, among which are a pitch-based approach to time delay estimation and a closed-form location estimator. The implementations concentrate on room oriented solutions - microphone arrays placed in room walls. The sound localization is applied in three dimensions but one of the dimensions (the vertical axis) is highly restricted. The reported accuracy of localization is very high - the space resolution is in the order of centimeters. However, it is impossible to compare to the accuracy of the work presented in this thesis, because the placement of sensors is different. In the case of room-oriented placement it is impossible to define a direction or distance from the array since the array surrounds the source. Furthermore the application domain of the room-oriented implementation is much more limited than that of a free standing array.

A model for 3D sound localization that uses both ITD and ILD is presented by Keith Martin of MIT in [19]. The work simulates the human auditory system and uses a binaural setup. Some assumptions and approximations of the real-life environment are made and as a consequence the system can only estimate angular direction but no distance. The experiments are performed in an idealized environment (e.g. noise is eliminated) and an angular accuracy of around 5° is achieved. All computations are performed offline.

One of the applications with hybrid methodologies is the work of Udo Bub, Martin Hunke and Alex Waibel at Carnegie Mellon University [8]. They use a second

modality (vision) to improve the performance of the localization device by refining the location estimate, provided by the sound localization. Their approach involves the use of ITD only as determined by a linear array of 15 microphones for redundancy. This setup limits their application to recognition in a single half-plane in front of the array. With acoustic means only, their system achieves an angular accuracy of around 5° and under 10% in distance with background noise only. This error is estimated from a single point in space in front of the array. The authors indicate that competing noise can significantly degrade the accuracy of localization.

Another work with a practical implementation is by Daniel Rabinkin et al. at Rutgers University [20]. Their system is limited to two-dimensional localization yet it uses two subarrays of 4 coplanar microphones each, placed on room walls. It employs a DSP to estimate ITD by using a cross-power spectrum phase algorithm. Then a space search algorithm minimizes the error between estimated and computed delays to produce a location estimate. The performance with the algorithms being run offline is reported as a percentage of deviation from a preset bound of 6° and is reported as being generally lower than 20%. The performance from the online tests is reported as being from “moderately well” to “quite poor”.

A number of other publications [3][11][12][13][14][15][16][17][25] treat theoretical aspects of auditory localization and the various methodologies used in practical sound source localization. However, there is no experimental work or actual implementations reported. The problem of full three dimensional sound source localization with a compact mobile structure, as examined by this thesis, has not been considered in any other work included in the scope of this survey.

Chapter 2

Theoretical problem and sensor structure

Required by versatile applications such as the dimensional hearing of a mobile robot, we cannot use room-oriented solutions [6][20], which typically use a large intersensor distance, with all the sensors fixed in the room. In our case the sound source will necessarily be located outside of the sensor structure. Furthermore the distance to the source will generally be significantly larger than the dimensions of the sensor structure. Most of the sound sources that can be of interest for the purposes of sound localization are compact enough to be assumed point-sources. If the source cannot be approximated by a point then the problem is different and for the case of a source size comparable to the source-detector distance, the problem is outside the scope of this work. The same applies to the case of distinct sources of comparable intensity. To determine the minimum number of sensors and their optimal placement, we need to look into the geometrical aspects of the problem.

2.1 Parameters

From the front of the spherical acoustic wave, the two main parameters that can be estimated are ITD and ILD. Assuming that the speed of sound is constant, which is true only for uniform media (density, temperature, chemical and physical contents, etc.), we can claim that ITD is proportional to the difference of the distances between each of the detectors and the sound source:

$$\text{ITD} \sim r_1 - r_2, \quad (2.1)$$

where r_i is the distance between the sound source and the i -th microphone, $i = 1, 2$, and \sim indicates proportional (Fig. 2.1). Also since the amplitude of the sound wave,

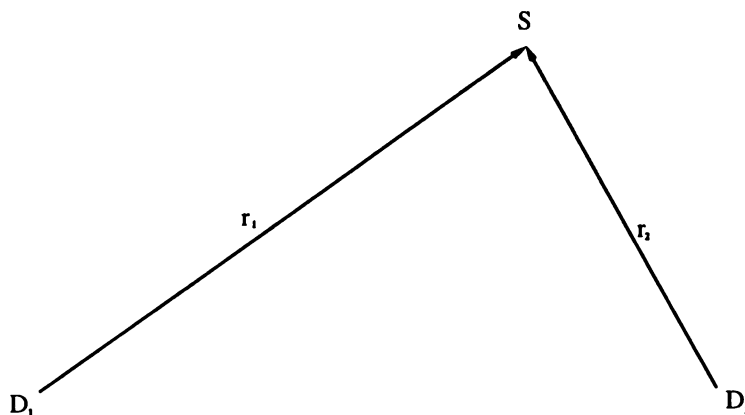


Figure 2.1: Two detectors and a sound source

or the intensity of the sound, varies inversely with the square of the distance, the ILD is proportional to the difference between the inverse values of the square of the distance. However if we take the difference we will be confronted with high-order terms in the equations which will lead to unnecessary complication of the computations. A much simpler form is provided by the ratio of the two values:

$$\text{ILD} \sim \frac{r_2^2}{r_1^2} \quad (2.2)$$

Both parameters can be measured and if we are able to unambiguously define the geometric representation then we would have a good method for solving our problem.

2.2 Number and placement of detectors

In order to determine the minimum number of detectors necessary we will first have to consider the geometric representation. For each couple of microphone detectors, we can derive the locus of the source points corresponding in three dimensional space to a given measured value of ITD or ILD. From equation (2.1) the locus is represented by a hyperboloid of two sheets of revolution with foci D_1 and D_2 . Depending on the sign of the difference one of the sheets contains the source location. Then from equation (2.2), for the matching ILD, it is less obvious but the forthcoming calculations will show the locus is a sphere [1][2][21]. The intersection of these surfaces defined by a number of detector couples will determine the ambiguity of the solution. It is clear that, apart from some special cases, with three couples the intersection is two points located symmetrically on both sides of the plane containing the three detectors. If four non-coplanar detectors are used the intersection is a single point. Because of the assumed isotropy of the locality, an equidistant structure seems reasonable. In the case of four sensors this suggests a tetrahedron (Fig. 2.2). A mobile robot requires that the structure of the sensor array be compact, while accuracy consideration requires a large array. In the experiment, an equal-side tetrahedron with a 20cm side was used.

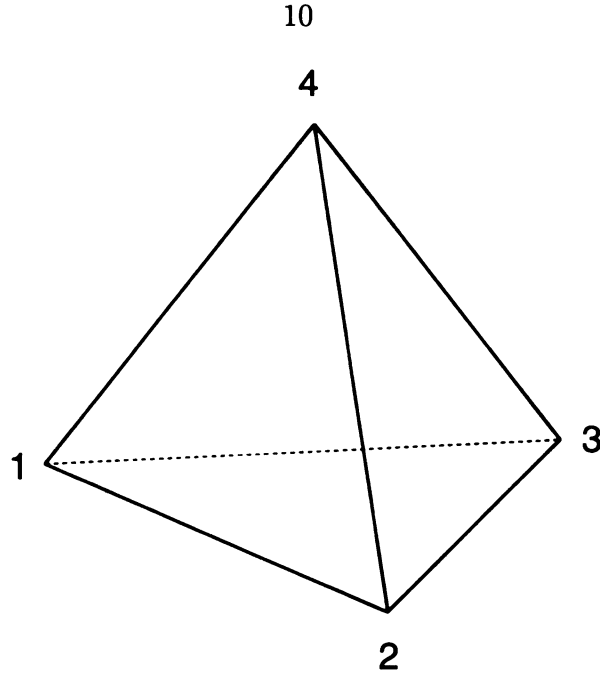


Figure 2.2: Microphone placement on the array

2.3 Geometric representation

As we established earlier in equations (2.1) and (2.2), we need only use the geometric distances r_i between the detectors D_{ij} and the sound source S . Without loss of generality we assume the following locations of the four detectors in a Cartesian coordinate system: $D_1(0, 0, 0)$, $D_2(2a, 0, 0)$, $D_3(0, 2a, 0)$, $D_4(0, 0, 2a)$. In this representation the detectors are not equidistant: the distance between D_1 and the rest will be $2a$ and the distance between any pair of D_2 , D_3 and D_4 will be $2\sqrt{2}a$ (Fig. 2.3).

2.3.1 From Intensity Ratios (IR)

We can define the following constants corresponding to the IRs:

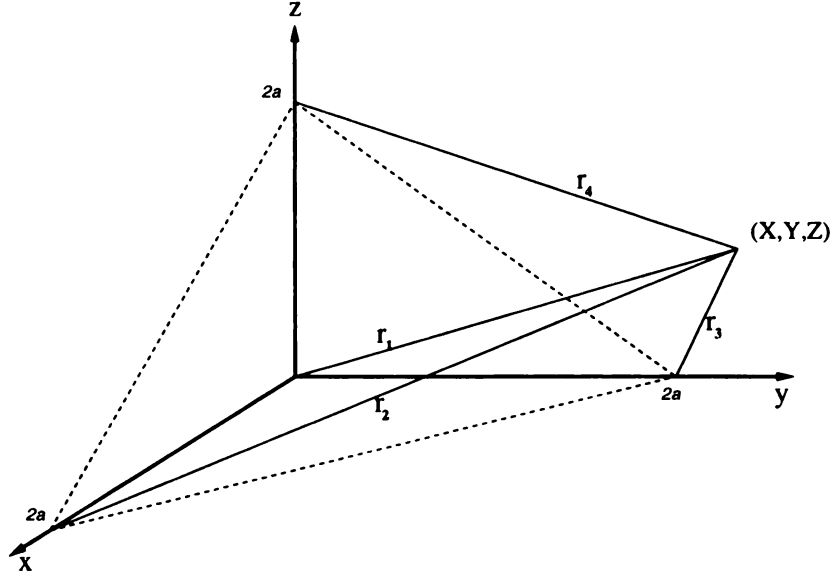


Figure 2.3: Distances from source to four detectors

$$c_{ij} = \begin{bmatrix} - & \frac{I_1}{I_2} & \frac{I_1}{I_3} & \frac{I_1}{I_4} \\ - & - & \frac{I_2}{I_3} & \frac{I_2}{I_4} \\ - & - & - & \frac{I_3}{I_4} \\ - & - & - & - \end{bmatrix} = \begin{bmatrix} - & \frac{r_2^2}{r_1^2} & \frac{r_3^2}{r_1^2} & \frac{r_4^2}{r_1^2} \\ - & - & \frac{r_3^2}{r_2^2} & \frac{r_4^2}{r_2^2} \\ - & - & - & \frac{r_4^2}{r_3^2} \\ - & - & - & - \end{bmatrix} \quad (2.3)$$

where I_i are sound intensities and the i are the respective detector numbers.

Using those we can write the following equations:

$$c_{12} = \frac{(x - 2a)^2 + y^2 + z^2}{x^2 + y^2 + z^2}, \quad c_{13} = \frac{x^2 + (y - 2a)^2 + z^2}{x^2 + y^2 + z^2}, \quad c_{14} = \frac{x^2 + y^2 + (z - 2a)^2}{x^2 + y^2 + z^2} \quad (2.4)$$

$$c_{23} = \frac{(x - 2a)^2 + y^2 + z^2}{x^2 + (y - 2a)^2 + z^2}, \quad c_{34} = \frac{x^2 + (y - 2a)^2 + z^2}{x^2 + y^2 + (z - 2a)^2}, \quad c_{24} = \frac{x^2 + y^2 + (z - 2a)^2}{(x - 2a)^2 + y^2 + z^2} \quad (2.5)$$

From (2.4) we get the following system:

$$\begin{cases} (x - L_{12})^2 + y^2 + z^2 = R_{12}^2 \\ x^2 + (y - L_{13})^2 + z^2 = R_{13}^2 \\ x^2 + y^2 + (z - L_{14})^2 = R_{14}^2 \end{cases} \quad (2.6)$$

where $L_{ij} = \frac{2ac_{ij}}{c_{ij}-1}$ and $R_{ij} = \frac{2a}{|1-c_{ij}|}\sqrt{c_{ij}}$, $i = 1$, $j = 2, 3, 4$.

And from (2.5) we get the following system:

$$\begin{cases} (x - L_{23})^2 + (y + L_{23})^2 + z^2 = R_{23}^2 \\ x^2 + (y - L_{34})^2 + (z + L_{34})^2 = R_{34}^2 \\ (x + L_{42})^2 + y^2 + (z - L_{42})^2 = R_{42}^2 \end{cases} \quad (2.7)$$

where $L_{ij} = \frac{2ac_{ij}}{c_{ij}-1}$ and $R_{ij} = \sqrt{2}\frac{2ac_{ij}}{|1-c_{ij}|}$, $i = 2, 3, 4$, $j = 3, 4, 2$.

Systems (2.6) and (2.7) consist of the equations of six spheres with radii R_{ij} and centers described by L_{ij} . Since a is a constant, by measuring the c_{ij} (which are a representation of the IRs) we can reconstruct the whole picture - the intersection of the 6 spheres will be our solution. It remains to show the uniqueness of that solution. By grouping the equations from (2.6) and (2.7) two by two and eliminating the quadratic terms we get a linear system of three equations with three unknowns, whose solutions are a superset of the solutions of the systems (2.6) and (2.7), i.e. if the system so obtained has a unique solution, so will the original system.

$$2 \begin{pmatrix} L_{12} & -L_{13} & 0 \\ L_{23} & -L_{23} & -L_{14} \\ L_{42} & L_{34} & -L_{34} - L_{42} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -R_{12}^2 + R_{13}^2 + L_{12}^2 - L_{13}^2 \\ -R_{23}^2 + R_{14}^2 + 2L_{23}^2 - L_{14}^2 \\ -R_{34}^2 + R_{42}^2 + 2L_{34}^2 + 2L_{42}^2 \end{pmatrix} \quad (2.8)$$

The determinant of (2.8) is:

$$\Delta = L_{12}L_{23}L_{34} + L_{12}L_{23}L_{42} + L_{13}L_{14}L_{42} + L_{12}L_{34}L_{14} - L_{13}L_{23}L_{34} - L_{13}L_{23}L_{42} \quad (2.9)$$

As we mentioned earlier the equations (2.8) must be consistent by definition so we are only interested in their independence. However, both will be ensured by having $\Delta \neq 0$. In our case we always have $r_i \gg 2a$ and therefore $c_{ij} \cong 1$, which means the L_{ij} are of the same order. Thus for layouts as the one used in this work (2.8) has a

unique solution and hence the same is true for (2.6) and (2.7).

We also have a purely geometrical indication of that fact, which is however difficult to visualize. Consider the 3 spheres described by (2.6). Their centers are non-collinear and their radii are of the same order. Therefore in the worst case two of them intersect in a circle and the third intersects that circle in two points. The same can be said about the spheres derived from (2.7). Furthermore we know the two new points are not collinear with the first two since the centers of the second set of spheres does not lie in the same plane as those from the first set. Hence only one point can be the intersection of all six spheres, which suggests the uniqueness of the solution.

As was mentioned above, in our case $c_{ij} \cong 1$. That was helpful (although not necessary) in establishing the uniqueness of the solution. Unfortunately that brings up a problem for the practical computing of the exact location of the sound source, because that means L_{ij} and R_{ij} grow rapidly as c_{ij} approaches 1. (The first derivative of L_{ij} and R_{ij} as functions of c_{ij} has large values as c_{ij} tends to 1, which means a large error in L_{ij} and R_{ij} when determined by c_{ij}). However the geometrical representation of the problem shows that we have an intersection of spheres with very large radii but whose centers are far apart, which indicates that the solution is stable.

Thus the solution is an intersection of a number of spheres. The measurement error impacts both centers and radii of the spheres but the error in radius is the one that determines the shape of the solid, representing the margin of errors. It is wide when viewed from the sensors and flat when viewed from the side. The actual impact of these observations, as it will clearly be shown from the experiments, is that the ILD, which are a representation of the IR, will provide a good indication of the distance to the sound source but a poor estimate of the angle.

2.3.2 From Phase Differences (PD)

So far we have only considered the IR. Although the PD seem more promising for the purpose of obtaining an accurate solution they are much more difficult to deal with analytically. Before we had the only quadric surface with spherical symmetry - the sphere - and thus we were not concerned with rotated axes: the equation of the sphere can only differ from its canonical form by a factor of translation, reflected as a constant term added to any of its variables. Unfortunately in the case of PD the locus of the possible solutions for each pair of detectors constitutes a hyperboloid of revolution of two sheets. Let us redo some of the steps we did for the IR. We define analogically some constants:

$$2d_{ij} = \begin{bmatrix} - & \varphi_1 - \varphi_2 & \varphi_1 - \varphi_3 & \varphi_1 - \varphi_4 \\ - & - & \varphi_2 - \varphi_3 & \varphi_2 - \varphi_4 \\ - & - & - & \varphi_3 - \varphi_4 \\ - & - & - & - \end{bmatrix} = \begin{bmatrix} - & r_1 - r_2 & r_1 - r_3 & r_1 - r_4 \\ - & - & r_2 - r_3 & r_2 - r_4 \\ - & - & - & r_3 - r_4 \\ - & - & - & - \end{bmatrix} \quad (2.10)$$

In a similar way we obtain the following system:

$$\begin{cases} \frac{(x-a)^2}{d_{ij}^2} - \frac{y^2}{a^2-d_{ij}^2} - \frac{z^2}{a^2-d_{ij}^2} = 1 \\ \frac{-x^2}{a^2-d_{ij}^2} + \frac{(y-a)^2}{d_{ij}^2} - \frac{z^2}{a^2-d_{ij}^2} = 1 \\ \frac{-x^2}{a^2-d_{ij}^2} - \frac{y^2}{a^2-d_{ij}^2} + \frac{(z-a)^2}{d_{ij}^2} = 1 \end{cases} \quad (2.11)$$

where $i = 1$, $j = 2, 3, 4$.

These three equations describe the three hyperboloids, with transverse axes designated by the terms with d_{ij}^2 in the denominator, and conjugate axes shifted by a in the positive direction. For the case of the remaining three couples a rotation factor intervenes and the hyperboloids are not described in their canonical form. It can be

determined that the rotation is of angle $\frac{\pi}{4}$ and there is some translation as well.

Again we obtain six quadratic equations with three unknowns and one measurable parameter, hence we can produce a solution in an empirical way. Those equations are inappropriate for showing the uniqueness of the solution but they can be used in combination with the equations from the IR and actually most of the time they are used by themselves (as the corresponding ITD) for sound source localization.

Again from the geometrical representation of the solution the margin of errors is a solid, which this time is oriented with the pointed side towards the detectors. It is obtained from the intersection of 6 hyperboloids and is almost impossible to visualize but intuitively it can be assumed to be elongated enough to suggest a considerable error in distance, while providing a very good estimate of the angle. This is the exact opposite of the case with the ILD which makes these two parameters complement each other very successfully and suggest their simultaneous use with sound localization systems that need to estimate the distance.

Chapter 3

Methodology

As outlined in Chapter 2 efficient sound localization can be performed after having extracted the necessary differential features. It was shown that the minimum number of detectors required to obtain unambiguously a solution in three dimensional space is four and that it is unique. In order to fully solve the problem of three dimensional sound localization two main steps need to be performed. First the chosen features need to be extracted from the acquired signal. Then using those features the actual sound location must be estimated.

3.1 Feature Extraction

As discussed, the two features considered in this work are ITD and ILD. Their extraction is based on the fact that the signal detected by the different sensors bears a significant degree of coherency when produced by a single dominating source.

Using the appropriate hardware, the acoustic signal can be first converted to electrical signal (microphones) and then to digital form (analog to digital converter board). The digitized data is a sequence of values representing the intensity of the sound, as picked by the respective detector for a determined period of time. A window

of sufficient duration is used to define the searchable domain. Some preprocessing is applied to ensure satisfactory quality of the sample. For instance, the amplitude of the signal's envelope can vary over time, e.g. with speech this corresponds to accents and pauses within and between words. These sound blanks contain little useful information and using them can degrade the quality of the estimates. In order to avoid this problem, the window is divided into smaller intervals in which the variation of the signal is evaluated (Fig. 3.1). This preprocessing selects only signals with high variance for feature extraction. A measure of the "efficiency" of the sample is returned by the procedure as the percentage of used subframes in the whole window.

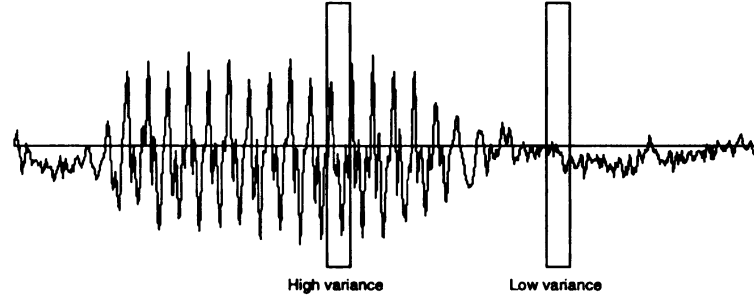


Figure 3.1: Preselection according to signal variance

The next phase involves the use of a cross-correlation procedure to determine the shift between the sampled signals at each of the sensor couples. This gives a direct measure for the ITD [8]. We find the peak of the cross-correlation function varying across a range of possible time-delays (3.1), when $j = 0$, k varies from 0 to $n - 1$ and when $k = 0$, j varies from 0 to $n - 1$:

$$R_{max} = \max_{\substack{j=0, 0 \leq k < n \\ k=0, 0 \leq j < n}} \left(\frac{\sum_{i=0}^{N-1} (X_{i+j} - \bar{X}_j)(Y_{i+k} - \bar{Y}_k)}{\sqrt{\sum_{i=0}^{N-1} (X_{i+j} - \bar{X}_j)^2} \sqrt{\sum_{i=0}^{N-1} (Y_{i+k} - \bar{Y}_k)^2}} \right) \quad (3.1)$$

where R_{max} is the maximum cross-correlation, X_i and Y_i , $i = 0, \dots, N$ are the samples from both channels, $N + n$ being the total number of samples in the window and n

the number of samples corresponding to half of the maximum possible time delay.

The time-shift, T will be proportional to the value of j or k that maximizes the value of 3.1, more precisely it is the product of that number and the digitization interval (the inverse of the sampling frequency). The value at the maximum is selected and is returned along with a parameter reflecting the sharpness of the correlation curve at the peak (Fig. 3.2). A combination of those two parameters is used as a quality estimate for the ITD (“score”). A high value of R_{max} is an indication of good coherence of the signals from both channels, which is true for a clear and compact sound source but also is true for an even isotropic noise. The second parameter, however, discriminates accurately between those two cases and helps select only the appropriate matches: a wide, flat correlation curve would indicate a predominance of noise, and a narrow, sharp curve - a clear, distinguishable sound source.

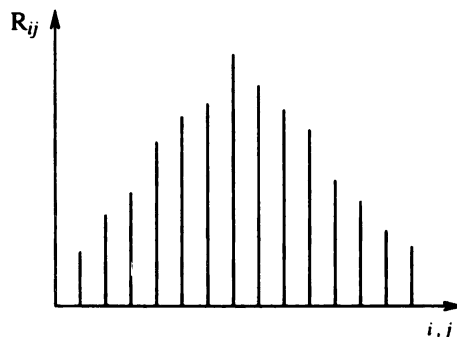


Figure 3.2: Typical cross-correlation curve with good sharpness

We should note a very useful side effect, which is closely related to the precedence effect in human auditory perception [13][14]. In case there are reflections of the incoming sound from incident surfaces (echoes, reverberation [11]) a secondary maximum will appear in the correlation curve. However, with the presented approach, it will be ignored because the coherency of that signal will be significantly lower and thus it will correlate less well (lower and wider peak).

Using the thus obtained information for the ITD, it is possible to evaluate the ILD

by computing an integral value of the signal energy from the shift-adjusted signal. The value for microphone pair 1 and 2 is shown in equation 3.2.

$$\text{ILD}_{12} = \int_{t_i}^{t_j} \frac{S_1(t)}{S_2(t+T)} dt \quad (3.2)$$

where $S_1(t)$ is the signal picked from microphone 1 and $S_2(t+T)$ is the signal picked from microphone 2, T is the previously determined time shift and $t_i - t_j$ is the length of the sample window.

The estimates for ITD and ILD are considered reliable only if the efficiency and score of the sample are satisfactory, i.e. above a predefined threshold. Thus the described procedure not only extracts the needed features but will also suggest when a sample can be reliably used for localization and when it should be discarded as useless.

3.2 Source localization

Once the ITD and ILD are extracted from the signal picked up by the detector array, the next step is to perform the actual sound source localization. The discussed disadvantages of the currently available methods can be avoided to a large extent by taking a learning-based approach. As stressed in Chapter 2 these features uniquely define a solution and thus we have a direct correspondence between the three dimensional coordinates of the sound source and the extracted ITD and ILD values. We also noted the extreme complexity of the actual mapping function. It is appropriate then to use learning as an efficient way of approximating complex high-dimensional functions [22]. Being able to explicitly model the function would significantly increase the accuracy of prediction. However, in the current case it was shown that when considering all the presented variables, it is very difficult to establish the form of such a model. Fur-

thermore, the model might strongly depend on the training environment and hence the generality of the performance application domain will be seriously limited.

In the current case the input feature space X is 12 dimensional, 6 for ITD and 6 for ILD, one for each combination of detector pairs. The output space Y is 3-dimensional: azimuth, elevation and radial distance. Thus the mapping is of the form $[x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{26}] \rightarrow [y_a, y_e, y_r]$. The polar representation in output space is used because it is known that the radial distance is a very unreliable estimate.

The goal of the SHOSLIF recursive partition tree (RPT) is to find the top k nearest neighbors in $O(\log(n))$ time, where n is the number of training samples learned and stored. The SHOSLIF scheme [22][23] automatically generates the RPT, which recursively partitions the input space X into smaller and smaller cells. Each leaf node contains exactly one training sample (x_i, y_i) , where $x_i \in X$ and $y_i \in Y$. The partition is based on automatically computed principal component subspace or linear discriminant subspace at each internal node of the RPT. A distance-based weighting according to (3.3) is applied to the y_i vectors of the top- k matched leaf nodes to give an interpolated output vector $y \in Y$, given input vector $x \in X$.

$$y = \frac{1}{\sum_{i=1}^k w_i} \sum_{i=1}^k w_i y_i \quad (3.3)$$

where w is the weighting function and y_0 is the nearest neighbor:

$$w_i = \alpha^{-\frac{\|y-y_i\|}{\|y-y_0\|+\epsilon}} \quad (3.4)$$

More detail about SHOSLIF is available in [22] and [23].

Chapter 4

Experimental setup and results

In order to test the methodology, an experimental setup was used to perform a number of tests. A set of four identical Lavalier microphones is placed at the tips of a solid tetrahedron with 20cm side (Fig. 2.2). The signal from the microphones is amplified by four modular microphone preamplifiers to bring the signal level in range. It is then supplied to an analog-to-digital converter board mounted in a personal computer. The software is designed to visualize, train, recognize and run various sound localization related tasks. Samples were taken from various points with known 3D coordinates, some were used for training and others for testing. The results were analyzed with linear search and the performance of SHOSLIF was evaluated.

4.1 Experiment and results

The dedicated hardware was built from low priced and widely available items to keep the project simple and economical. All experiments were held in the Pattern Recognition and Image Processing laboratory of the Department of Computer Science, which is hardly suitable for high precision acoustic experiments. The test space is located in the middle of the laboratory, in between cubicles with computers and

reflecting, and absorbing surfaces of irregular shape. The number of devices producing strong noises of different frequencies and levels is above 20. Often times human speakers would be active in the background while taking samples for training or retrieval. This highly problematic environment was actually intended in order to simulate the real world environments, in which a typical sound localization device would be intended to work.

4.1.1 Hardware

Some preliminary tests with low quality microphones emphasized the need for preserving the inherent coherency of the incoming signal. This led to the introduction of better quality Lavalier microphones, made for high fidelity voice recording. The choice fell on Audio-Technica ATR35s condenser microphones. Their polar pattern is omnidirectional, the frequency response is 50 – 18000 Hz, the sensitivity is $-54\text{dBm} \pm 3\text{dB}$, 1kHz at 1Pa, impedance of $1000\Omega \pm 30\%$. The array structure was built from cardboard and the microphones taped to the edges.

The signal from the microphones is preamplified by four inline modular microphone preamplifiers of the stick-on series by Radio Design Labs, model STM-2. Their input impedance is $5\text{K}\Omega$ unbalanced, the output impedance is 150Ω , the output level is up to $+4\text{dB}\mu$, the frequency response is $50\text{-}25000\text{Hz} \pm 1\text{dB}$, the gain is adjustable from 0 to 65dB, the THD is under 0.05% and the noise is under -80dB bellow $+4\text{dB}\mu$ with the microphones used. With those preamplifiers the signal from the microphones is taken to the $\pm 0.1 - 1$ V peak-to-peak range suitable for digitizing by the convertor.

The next component is the analog to digital converter board. It is the most expensive item in the whole setup. The one used in this work is made by Computer Boards and is a 12 bit, 16 single-ended channel, programmable gain ADC with 160kHz

maximum aggregate sampling frequency. Its accuracy is 0.01% of the reading ± 1 bit, the minimum current is 250nA and the input impedance is 10M Ω . The minimum conversion time is 3.3 μ s and the linearity is ± 1 bit. It is used to digitize the input on four of its channels at the rate of 16kHz per channel. This rate is sufficient to capture the second harmonic of the highest human-made tone of 3000Hz. The maximum available rate of the board, which would be 40kHz per channel, is not used because the higher frequencies contain predominantly noise, but could be easily implemented if necessary.

The computer used is a stock IBM-compatible PC, based on a 200MHz Intel Pentium MMX CPU. The operating system is Microsoft Windows 95, imposed by ADC board driver availability.



Figure 4.1: Training the system

4.1.2 Experiment

At the training stage a continuous sound, originally produced by a human speaker uttering a short sentence, is reproduced using a hand held tape recorder, from a set of previously defined locations (Fig. 4.1). Without significant loss of generality, the span of the training grid was set to an arbitrary section of $3 \times 3 \times 2.1$ meters, with the microphone array in the middle of one of the sides. The density is linear in Cartesian coordinates with a granularity of 0.3m. However, only 237 of the thus defined 700 points were used for training. They were selected to simulate a uniform angular density and ten samples were taken from each of those points. Thus the angular span of the training area was about 180° in azimuth and a little less than that in elevation. The site was the PRIP Laboratory (Fig. 4.2) where the acoustics is very challenging and the background noise was significant. At this time a room with better acoustic properties (e.g. anechoic chamber) was not available but as it was mentioned earlier, this type of environment is close to the one in which an actual sound localization device would be exposed to.



Figure 4.2: Experimental setup

At the recognition stage the same device produced the original sound, however the performance of the system was tested with other sources, as well. A human speaker

would produce variable results. One important observation was that the quality of the sound, which directly influences the reliability and the accuracy of recognition, depended on the amount of useful information (ratio of sounds vs pauses) but mostly on the compactness of the source - the aperture of the mouth.

The system can be started to collect 0.5s samples and provide location estimates based on each of them or to store them for further analysis. Test samples from 79 points on the grid, but different from the ones used for training, were taken. They were analyzed offline with the specially designed software. Estimates for the location of each of the test points were thus produced and recorded. They were compared to the known actual values of the three dimensional coordinates and the error was computed as the difference between the actual and estimated value for the angles, and the ratio of that difference to the actual value, for the radial distance.

The employed algorithm uses two parameters for fine tuning. One is the relative weight of the two extracted features: ITD and ILD. Because of the importance of the ITD, only the ILD was multiplied by a variable factor, called *scaling on ILD*, thus increasing its weight (the original values being very low numbers) as needed. This allows us to estimate the relative influence of those two parameters on the accuracy of the results. A low value of this parameter would mean neglecting the ILD (a value of zero means only ITD are used), while a higher value indicates a predominance of the ILD. Their relative weight is practically equal for a value of around 13. The other parameter is the weight coefficient α in the interpolation of the retrieved nearest neighbors (3.4). A low value of α would indicate that all considered nearest neighbors are equally weighted (for $\alpha = 1$ we have averaging) while a big value of alpha emphasizes the role of the nearest neighbor.

It is known that ITD and ILD are frequency dependent, e.g. ILD uses predominantly the low frequencies, while higher frequencies are the only ones that can be used

for estimating the ITD. A preliminary signal filtering can be employed to leave only the useful frequencies when determining each of those two parameters. The actual response of those two filters can be another subject for fine tuning. However, the real-time implementation requirements for this project impose serious limitations on the amount of preprocessing that can be performed and thus spectral analysis of the signal is abandoned at this stage.

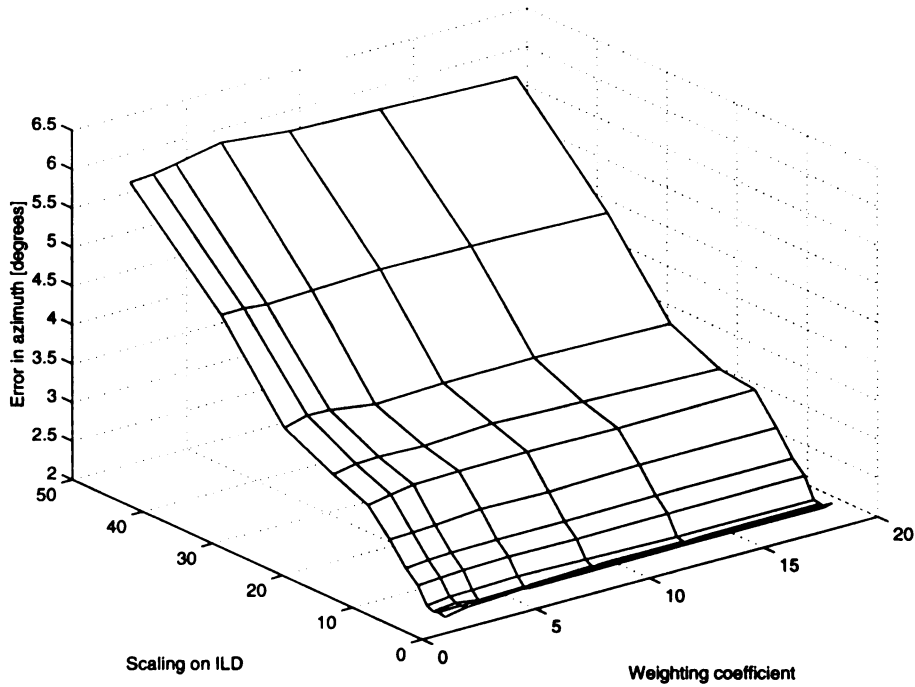


Figure 4.3: Distribution of error values for Azimuth

4.1.3 Results

The results obtained in this manner were used to study the above mentioned relations. A number of plots is used to show some observed trends. Fig. 4.3 shows how the relative weighing between ITD and ILD affects the accuracy of estimation of the azimuth, Fig. 4.5 - of the elevation and Fig. 4.7 - of the distance. The respective standard deviations are shown on Fig. 4.4, Fig. 4.6 and Fig. 4.8. The horizontal axes

are the scaling on ILD - the coefficient by which ILD is multiplied when considered for estimating the nearest neighbors, and the weighting coefficient α , which indicates how much influence the further neighbors have in the interpolation (see equation 3.3). The range on the axes was chosen equal on all plots for compatibility and the direction of the axis, representing the scaling on ILD, was inverted for the distance plot so the surface can face the viewer. The distance plot shows a trend of a descending error value but its further extent was not followed because the standard deviation is increasing for those same values, rendering the low error values unreliable. In these trials a number of KNN=7 (nearest neighbors) was used. The values of ILD are theoretically unbound hence it is impossible to get a correct number for the balance of relative weights of ITD and ILD but an empirically estimated value of scaling on ILD of around 13, for which their weight is approximately equal, was found.

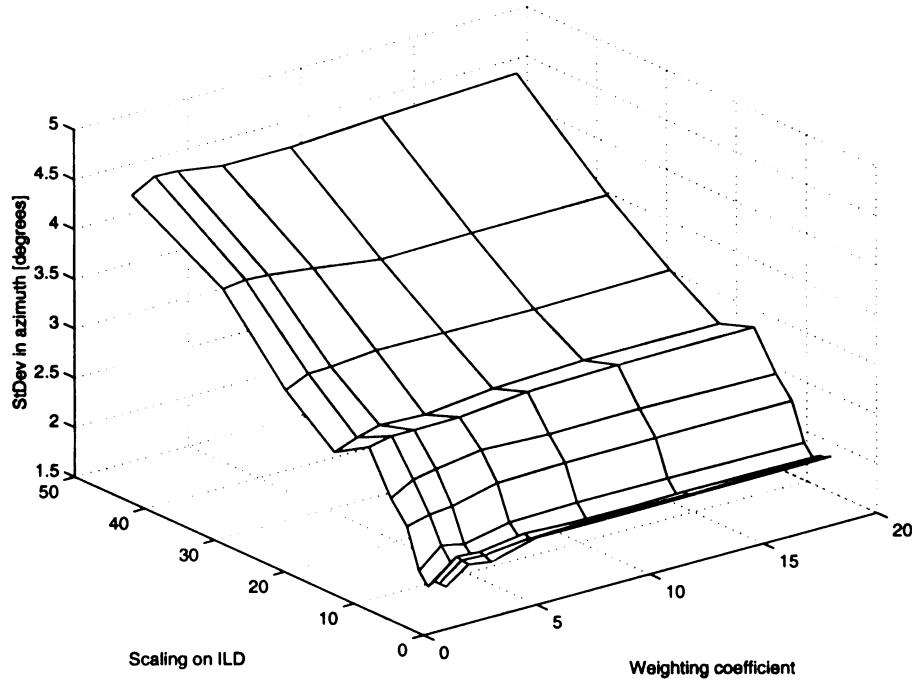


Figure 4.4: Distribution of standard deviation for Azimuth

The best precision measured for points located within the sector designated for

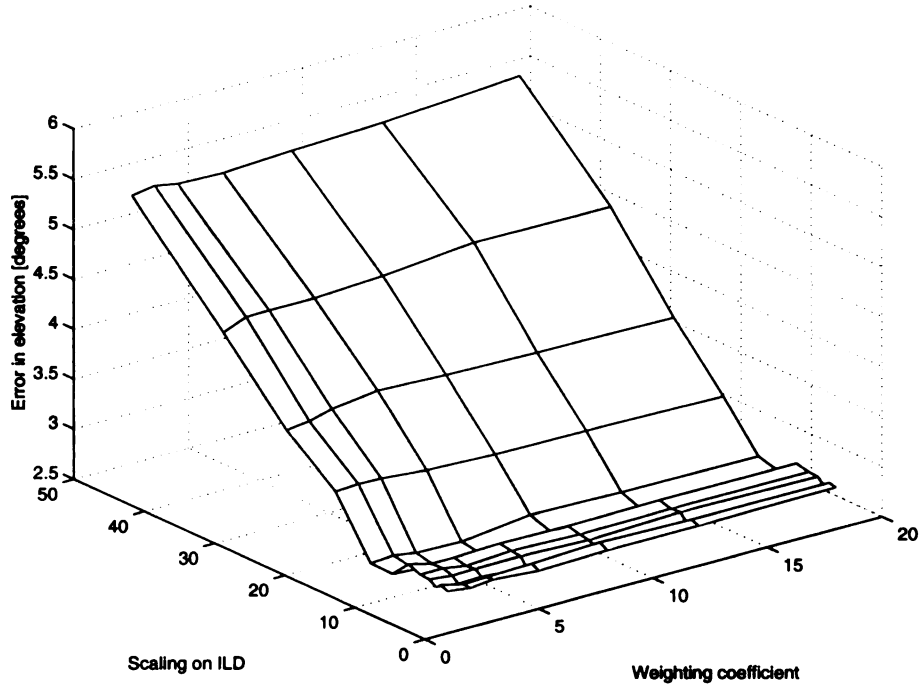


Figure 4.5: Distribution of error values for Elevation

training but between the grid points used for training, was estimated at around $\pm 2.2^\circ$ in azimuth, $\pm 2.8^\circ$ in elevation and $\pm 19\%$ in distance. Table 4.2, Table 4.4 and Table 4.6 show the error results for different values of the above tuning parameters. The respective values for standard deviation are shown in Table 4.3, Table 4.5 and Table 4.7. The superresolution is achieved by the KNN interpolation while the lower performance in distance is expected for such source - sensor placement. It should also be noted that the specific task the system was tested for - indicating the actual location of the sound - is very difficult for humans in such situations, as well.

Table 4.1: Comparative timings of various routines

PreProcessing	Linear Search	SHOSLIF
230ms	15ms	2.5ms

A sample set of error values, used to produce a single point in the average errors plot is presented in Table 4.8.

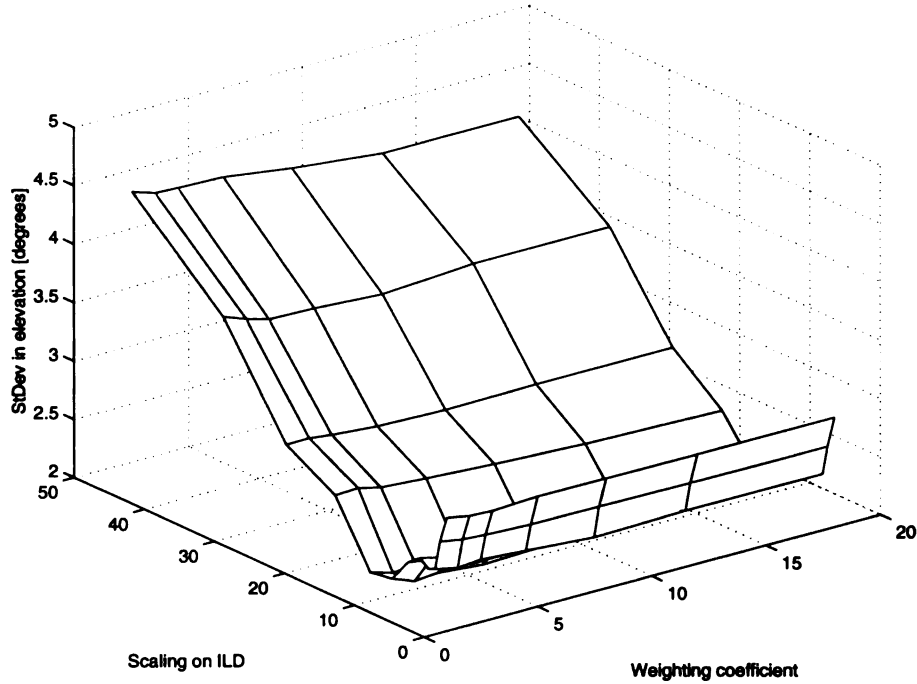


Figure 4.6: Distribution of standard deviation for Elevation

4.2 Program performance and details

For accuracy testing and parameter fine tuning a deterministic linear search algorithm was implemented to find the nearest neighbors in input space. The results obtained in such a way were used to estimate the performance of the SHOSLIF procedure. The speed was confirmed to be considerably faster with SHOSLIF, even for the low dimensionality of the input: only 12. As timed on the test PC, a single retrieval from the tree, with 2370 test samples, took 2.5ms on average for SHOSLIF, versus 15ms with the linear search (see table 4.1). The accuracy was comparable to that of the linear search. The timing for the preprocessing indicated an average of 230ms which, although being considerably longer than the retrieval time, is still shorter than the signal scan time of 500ms (single window). In other words the algorithm, as implemented, is twice as fast as needed for real time application.

The program was written in C++ and is object oriented with the exception of the

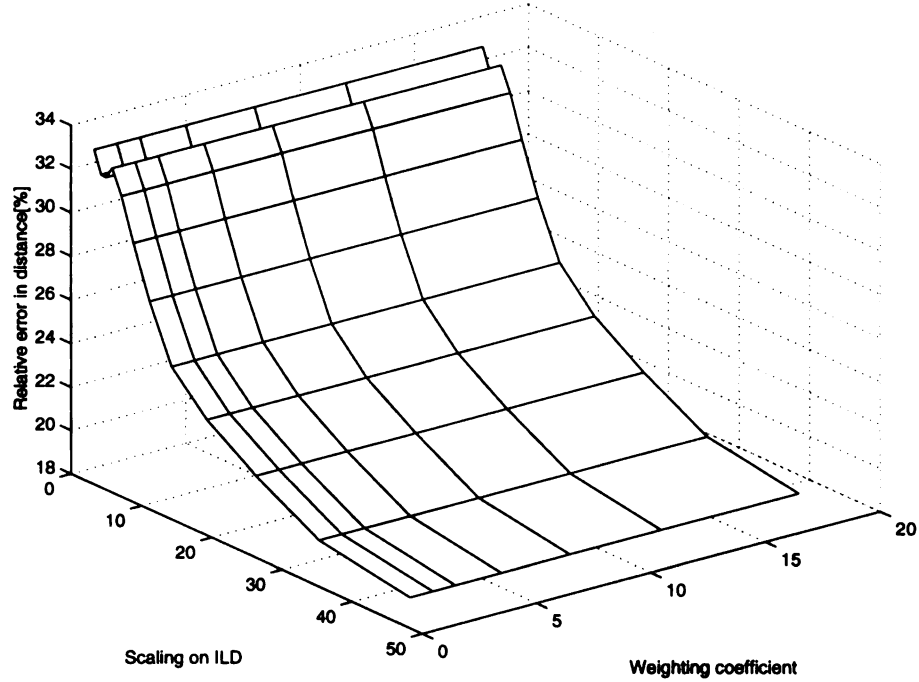


Figure 4.7: Distribution of error values for Distance

C code, adapted from a previous implementation of SHOSLIF. The Graphical User Interface allows the user to pass all necessary parameters to the program, to select the various options, as well as to view the waveform of the scanned signals (Fig. 4.9).

4.3 Implementation restrictions

As mentioned before, the system performed well despite different unfavorable factors, like background noise, reflections, unreliable sound sources, etc. However, it should be noted that although no exact measurements have been performed, these and some other factors would influence its reliability and accuracy depending on their strength. In most experiments the acoustic noise was kept at a S/N ratio of around 20dB (as estimated visually from the displayed waveform) but in real life situations the S/N ratio can be as high as 0dB (noise is as strong as signal). Another problem

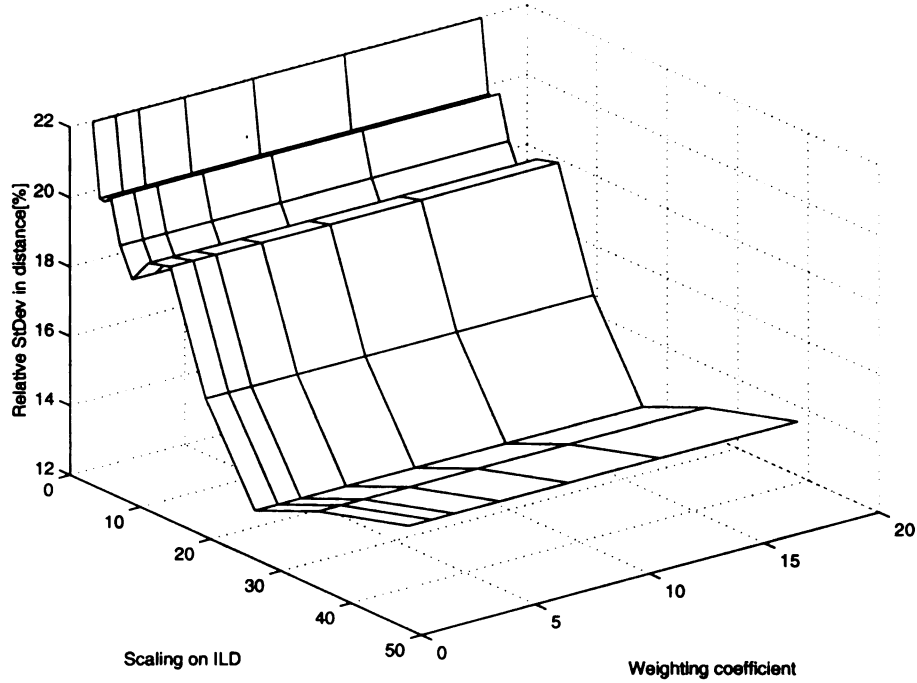


Figure 4.8: Distribution of standard deviation for Distance

would be multiple sound sources. In the case of signal reflection, the intensity of the reflected signal would be significantly weaker and thus it will be ignored by the preprocessing routine. However with secondary sources, the intensity of the sources can be comparable and this might lead to jumping between the two sources and even complete wash-out of the correlation curve and thus incorrect localization.

Most of the experiments were performed with a sound source steadily fixed in space. A moving source would present a challenge for the current implementation. With the current windowing of 0.5s a source movement would be equivalent to having a source of a larger size (aperture), which would produce a lower signal correlation. The performance with a shortened window has not been studied extensively at this point. In a similar way an influence on the accuracy of detection was observed when varying the size of the aperture of the sound source. For instance sounds produced with a wide open mouth would yield a high error value. An accurate study of this

Table 4.2: Average error in azimuth for different values of *Scaling on ILD* (x) and *weighting coefficient* (y) as plotted in Figure. 4.3 [degrees]

x/y	1	2	3	5	8	12	18
0	2.21	2.26	2.28	2.32	2.33	2.33	2.34
1	2.23	2.24	2.25	2.23	2.23	2.24	2.24
1.7	2.21	2.25	2.20	2.22	2.24	2.24	2.26
2.6	2.25	2.29	2.30	2.31	2.29	2.27	2.27
3.9	2.46	2.48	2.50	2.51	2.50	2.51	2.53
5.6	2.61	2.64	2.65	2.66	2.66	2.68	2.68
7.9	2.89	2.90	2.91	2.96	2.95	2.98	2.98
11	3.20	3.26	3.31	3.32	3.34	3.32	3.35
16	3.39	3.45	3.46	3.44	3.50	3.46	3.41
23	3.70	3.79	3.78	3.69	3.73	3.74	3.70
32	4.78	4.79	4.76	4.78	4.82	4.81	4.75
45	5.96	5.98	6.03	6.15	6.06	6.03	5.98

relation needs to be performed in order to determine the correct way of compensating the increase in source size.

One of the main disadvantages that training presents is the difficulty for a learning system to perform in unknown environments, compared to the environment in which it was originally trained. No exact measures have been taken to establish the actual error values in a new environment but many observations indicate that the system can perform within some reasonable expectations.

Another obvious limitation is the absolute distance from the detectors to the sound source. Because of the physical characteristics of sound propagation in the air, the coherency of the sound waves decreases significantly with distance: non-linear frequency absorption, distortion, fading, are just a few of the factors. A good example is the noise, coming from the engines of airplanes high in the sky - it is very difficult to establish the location of the source, even after turning around several times (another factor intervenes here, too: the relatively slow speed of sound introduces a serious delay, relative to the optical image, i.e. the actual location of the object). For

Table 4.3: Standard deviation of error in azimuth, for different values of *Scaling on ILD* (x) and *weighting coefficient* (y) as plotted in Figure. 4.4 [degrees]

x/y	1	2	3	5	8	12	18
0	1.93	2.09	2.06	2.18	2.18	2.18	2.19
1	1.93	2.10	2.09	2.14	2.15	2.15	2.15
1.7	1.93	2.12	2.10	2.17	2.13	2.15	2.15
2.6	1.85	2.03	2.01	2.11	2.10	2.09	2.09
3.9	1.97	2.16	2.11	2.21	2.22	2.22	2.21
5.6	2.36	2.42	2.44	2.55	2.57	2.57	2.58
7.9	2.57	2.70	2.74	2.81	2.78	2.79	2.77
11	2.99	3.04	3.03	3.04	3.11	3.14	3.15
16	2.77	2.86	2.93	2.91	2.99	3.02	3.03
23	3.18	3.28	3.29	3.34	3.33	3.32	3.34
32	3.91	3.94	3.93	3.87	3.78	3.80	3.81
45	4.44	4.57	4.56	4.49	4.49	4.55	4.61

the presented implementation distances of just 5 to 10 meters would already pose a serious problem.

Table 4.4: Average error in elevation for different values of *Scaling on ILD* (x) and *weighting coefficient* (y) as plotted in Figure. 4.5 [degrees]

x/y	1	2	3	5	8	12	18
0	2.92	2.89	2.92	2.88	2.90	2.90	2.91
1	2.95	2.90	2.91	2.92	2.95	2.93	2.93
1.7	2.91	2.88	2.89	2.85	2.88	2.88	2.87
2.6	2.96	2.93	2.94	2.94	2.92	2.96	2.94
3.9	2.96	2.97	2.93	2.96	2.94	2.95	2.94
5.6	3.01	2.96	2.96	2.97	2.97	2.98	2.97
7.9	2.84	2.83	2.78	2.78	2.83	2.80	2.81
11	2.85	2.87	2.83	2.82	2.90	2.88	2.87
16	3.41	3.42	3.41	3.37	3.34	3.34	3.31
23	3.79	3.82	3.88	3.94	3.91	3.89	3.87
32	4.48	4.57	4.58	4.58	4.62	4.71	4.69
45	5.43	5.47	5.43	5.41	5.45	5.49	5.59

Table 4.5: Standard deviation of error in elevation, for different values of *Scaling on ILD* (x) and *weighting coefficient* (y) as plotted in Figure. 4.6 [degrees]

x/y	1	2	3	5	8	12	18
0	2.98	2.94	2.93	2.95	2.94	2.94	2.94
1	2.74	2.71	2.70	2.68	2.67	2.67	2.66
1.7	2.49	2.45	2.47	2.44	2.39	2.41	2.41
2.6	2.46	2.42	2.41	2.41	2.42	2.38	2.39
3.9	2.52	2.48	2.47	2.45	2.46	2.45	2.45
5.6	2.44	2.41	2.37	2.38	2.36	2.35	2.33
7.9	2.24	2.16	2.18	2.14	2.16	2.15	2.14
11	2.21	2.15	2.17	2.13	2.15	2.17	2.15
16	2.73	2.74	2.69	2.66	2.63	2.59	2.56
23	2.97	2.97	2.95	2.92	2.90	2.91	2.91
32	3.82	3.75	3.69	3.68	3.65	3.70	3.69
45	4.53	4.47	4.46	4.45	4.37	4.29	4.29

Table 4.8: Error values for *Scaling on ILD = 11* and *weighting coefficient = 1*

Azim [°]	Elev [°]	Dist [%]	A [°]	E [°]	D [%]	A [°]	E [°]	D [%]
2.0	6.0	17.45	1.0	1.0	44.25	1.1	0.3	22.16
1.1	6.4	7.70	1.7	5.9	38.35	1.4	1.0	19.54
1.0	1.0	39.46	3.0	2.0	34.02	4.9	0.6	6.95
12.0	0.0	0.00	1.0	4.0	40.24	3.0	3.0	29.82
4.0	2.0	11.82	1.0	4.0	40.24	8.9	3.3	31.68
0.0	10.0	25.37	1.1	8.0	14.55	0.0	2.1	22.08
2.6	3.3	17.22	2.0	2.0	19.14	7.0	2.0	56.67
0.0	0.7	25.35	6.7	0.9	17.04	9.0	1.0	45.56
3.0	4.0	23.95	5.0	2.0	19.79	2.0	3.0	30.86
3.0	4.0	23.95	1.7	3.9	26.79	6.6	4.0	29.94
0.0	1.1	24.45	5.7	1.1	7.72	3.0	2.0	41.13
3.0	0.9	14.07	0.9	4.4	2.83	2.0	3.0	30.86
1.4	2.9	11.51	7.7	2.9	16.54	0.0	0.0	3.89
1.3	0.4	11.64	5.3	3.1	1.98	0.0	0.0	50.34
0.1	2.7	10.45	3.1	1.1	9.45	6.4	5.9	0.61
11.0	3.0	32.09	7.0	0.0	8.99	5.4	1.3	22.39
0.0	5.3	3.38	2.4	5.6	7.70	0.7	0.9	4.52
9.0	1.0	45.56	3.6	3.1	7.05	3.0	1.0	39.36
2.0	9.0	37.70	0.0	5.1	25.36	0.9	4.6	13.76
0.3	5.1	51.18	3.0	0.0	26.72	2.4	2.3	13.80
3.0	2.0	41.13	5.1	8.4	17.51	0.7	4.4	13.60
0.3	1.7	22.14	0.1	1.0	4.81	1.0	4.0	23.59
9.0	4.1	57.37	2.3	2.3	6.60	4.0	0.3	14.68
1.1	2.4	69.98	0.9	3.3	46.99	6.0	3.0	12.55
1.1	4.6	19.53	8.0	3.0	22.56	4.7	2.0	7.40
1.0	1.0	44.25	1.4	0.3	1.99	4.6	1.1	12.31

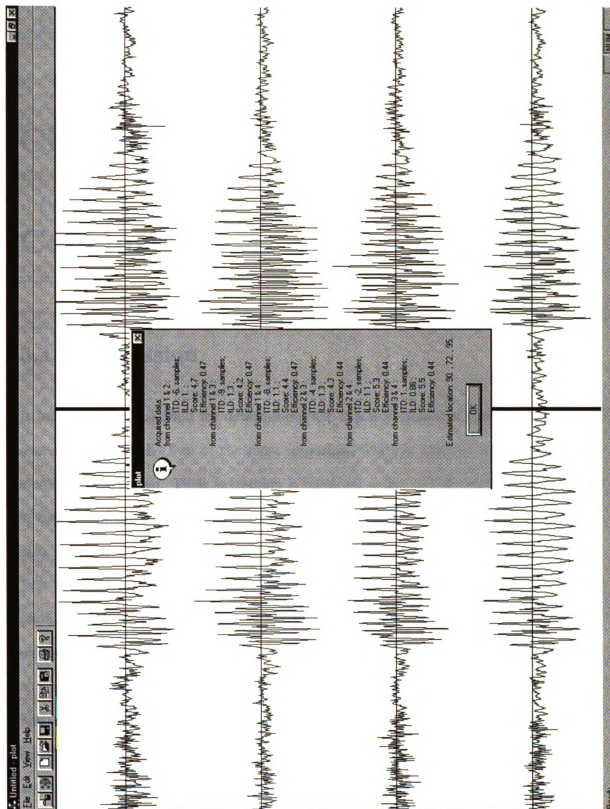


Figure 4.9: Screen with signals and data dialog

Chapter 5

Conclusions

5.1 Discussion

A learning-based method for determining the three-dimensional location of a sound source is presented. Two of the most important features used by humans in sound localization are used. Their extraction is based on a fast and efficient algorithm that is capable of not only computing those parameters with satisfactory accuracy but also provides a very useful means of evaluating the usability of the taken sample. The three dimensional localization is performed by a learning technique. The applicability of the proposed implementation is more general than the majority of the currently available solutions, in that various features can be used without the need to explicitly model the relationship between feature values and the 3D location estimates. The method needs to store a large number of samples (over-learning is avoided by SHOSLIF by only storing samples that are necessary). In order to achieve good accuracy the training density needs to be close to the expected resolution. This can lead to the need of taking samples from hundreds of three dimensional locations, and to ensure stability, several samples from each point need to be taken. Thus the total

number of training samples can some times approach tens of thousands. However, the logarithmic retrieval makes the system easily reach real-time response speed.

The originality of this work is in the versatility of its application domain. First the lack of spatial constraints allows for a wide range of applications. The use of a compact sensor array makes it suitable for mobile robots, embedded devices and other human-machine interaction apparatus. The simultaneous use of ITD and ILD as complementary attributes is another advantage because of their complementary character. It is made possible by the learning approach, also unique for this range of problems. The employed non-coplanar array with a minimal number of sensors is another distinctive feature of this work.

5.2 Future research

Some results suggest several directions for future work. The observed dependence of the “quality” of the sound on the size of its source (aperture) is an important issue because of the typical application domain of this method - human voice. Because of the different size of the mouth of different people and because of the inherent variations in the ways of pronouncing phonemes, the detectability of human sources can suffer severely. The chosen approach in preprocessing of captured sounds could be revised to reflect the expected variations in the source size.

Another direction for further study is the impact of sound source movement over the selected methodology. The relation between the speed of source movement and the discretization interval (the time segment, used to extract the features) can be adjusted by shortening the scanning time so that the source movement remains relatively slow but this approach will inevitably result in loss of accuracy. Other techniques for compensating the source movement can be developed to efficiently handle this

problem. For instance a sliding window algorithm can be employed, instead of a stop-and-go approach, to constantly monitor the available sound and adjust to its spatio-temporal variations.

The issue of multiple sound sources also should be studied in more details. With the present approach the presence of a second sound source is ignored but it degrades the quality of localization for the primary source.

It is also necessary to establish the degradation of performance when the system is put in an unknown environment. When training the system no assumptions are made about the acoustics of the environment and the ability of the system to overcome obstacles like noise, reflections, etc., could be due to the specific of training. It is interesting to establish a relation between the characteristics of the environment and the quality of sound localization.

It was mentioned that the generality was not lost when selecting a sector of the 3D space for training, although the system is claimed to be able to perform in all of the surrounding space. It remains to verify that claim experimentally but most important of all is the issue of the necessity to train in full 3D space around the array. It could be logical to expect that the data from a sector of space can be used to extrapolate the scope of localization to the full 3D space surrounding the system.

APPENDICES

Appendix A

Class “IADiff”: Signal

Preprocessing

This Appendix contains the C++ declarations and definitions of the class *IADiff*, which performs the preprocessing of the captured signal and outputs the ITD and ILD values, along with the quality measures.

Declarations

```
/*
Class IADiff: "Inter-Aural Differences"
Takes an array of samples, containing data from 2 or more channels.
Returns the Interaural Time and Level Differences along with two quality parameters:
    1) ratio of used samples (regions with high variance) to total number of samples;
    2) an arbitrarily determined score, indicating the certainty in determining the
        ITD & ILD from the cross-correlation.
It employs a fast cross-correlation implementation based on a heuristic search of
usable portions of the data.
*/

#ifdef IADIFF
#define IADIFF

#include <math.h>
#include <stdlib.h>

class IADiff
```

```

{
private:

    short *frame1, *frame2, *subframe;
    unsigned short maxShift;
    unsigned short prescanWindow;
    double *subVar;
    double bestShift, bestIratio, meanVar;
    unsigned short nSamples, nSubFrames;

public:

    // default constructor
    IADiff(void);

    ~IADiff(void);

    //AGC      first channel      second channel      # of samples/channel  number of subframes
    double Init(const unsigned short, const unsigned short, const unsigned short, const unsigned short,
    // array with data samples
    const unsigned short *);

    //ret:% used data ITD      ILD      score
    double IADcompute(double& , double& , double& );

private:

    //          frame 1      frame 2      number of samples/ch  size of XC window
    short MaXCorr(const short[] , const short[] , const unsigned short&, const unsigned short&,
    //sigma factor  max R      ratio  s-threshold
    unsigned short&, double&, double&, const double&);

    //ret:variance array w/data
    double PreScan(const short X[]);

    //ret:avg var.  variance
    double Variance(double[]);

};

#endif

```

Definitions

```

#include "IADiff.h"

IADiff::IADiff()
{
    const unsigned short Channels = 4;

    nSubFrames = 0;
    prescanWindow = 0;
    frame1=NULL;
    frame2=NULL;
    subframe=NULL;
    subVar=NULL;
    maxShift = 12;
}

IADiff::~IADiff(void)
{

```

```

    delete frame1, frame2, subframe, subVar;
}

double IADiff::Init(const unsigned short Ch1, const unsigned short Ch2, const unsigned short nSamples,
                   const unsigned short nSubFr, const unsigned short *Data)
{
    const unsigned short Channels = 4;
    unsigned int overflows1 = 0, overflows2 = 0, uprange1=0, uprange2=0;
    double modfactor1, modfactor2;

    nSubFrames = nSubFr;
    prescanWindow = unsigned short(nSamples/nSubFrames);
    frame1 = new short [nSamples];
    frame2 = new short [nSamples];
    subframe = new short [prescanWindow];    // a buffer for storing the subframes
    subVar = new double [nSubFrames+2];      // holds the variance of each subframe + MAX & MIN
    maxShift = 12;                          // the value of the maximum possible ITD

    for(int i=0; i<nSamples; i++)
    {
        frame1[i]=short(Data[Channels*i+Ch1]-2047);
        if(frame1[i]==-2047)
            overflows1++;
        else if(frame1[i]==2048)
            overflows1++;
        if(abs(frame1[i])>205)
            uprange1++;
        frame2[i]=short(Data[Channels*i+Ch2]-2047);
        if(frame2[i]==-2047)
            overflows2++;
        else if(frame2[i]==2048)
            overflows2++;
        if(abs(frame2[i])>205)
            uprange2++;
    }
    if(uprange1<0.01*nSamples && uprange2<0.01*nSamples)    // if there are <1% samples with amplitude >10%
        return(-1.0);    // of ADRANGE - signal to increase gain
    modfactor1=overflows1/double(nSamples);
    modfactor2=overflows2/double(nSamples);
    if(modfactor1 >= modfactor2)
        return(modfactor1);
    else
        return(modfactor2);
}

short IADiff::MaxCorr(const short X[], const short Y[], const unsigned short& N, const unsigned short& n,
                     unsigned short& sigma, double& Rmax, double& ratio, const double& sThreshold)
{
    long *rjo, *rok;
    double *meanX, *meanY;
    __int64 *StdX, *StdY;
    double *Rjo, *Rok;
    long maxC=0;
    long i,j;

    meanX=new double [n];
    meanY=new double [n];
    double sumX, sumY;
    rjo=new long [n];
    rok=new long [n];
    StdX=new __int64 [n];
    StdY=new __int64 [n];
    Rjo=new double [n];
    Rok=new double [n];

    // Computing the means of the two inputs

```

```

meanX[0]=X[0];meanY[0]=Y[0];
for (i=1; i<N; i++)
{
    meanX[0]+=X[i];
    meanY[0]+=Y[i];
}
meanX[0]=meanX[0]/N;
meanY[0]=meanY[0]/N;

for (j=1; j<n; j++)
{
    meanX[j]=meanX[j-1] + double(X[N-1+j] - X[j-1])/N;
    meanY[j]=meanY[j-1] + double(Y[N-1+j] - Y[j-1])/N;
}

//Computing the total "energy" of the signals
sumX=double(abs(X[0]));sumY=double(abs(Y[0]));
for (i=1; i<N; i++)
{
    sumX+=double(abs(X[i]));
    sumY+=double(abs(Y[i]));
}
sumX=sumX/N;
sumY=sumY/N;

ratio=sumX/sumY;

//Computing the cross-products "ro"
for (j=0; j<n; j++)
{
    rjo[j]=0; rok[j]=0;
    for (i=0; i<N; i++)
    {
        rjo[j]+=X[i+j]*Y[i];
        rok[j]+=X[i]*Y[i+j];
    }
    rjo[j]-= long(N*meanX[j]*meanY[0]);
    rok[j]-= long(N*meanX[0]*meanY[j]);
}

//Computing the standard deviation "sigma"
StdX[0]=0; StdY[0]=0;
for (i=0; i<N; i++)
{
    StdX[0]+=long((X[i]-meanX[0])*(X[i]-meanX[0]));
    StdY[0]+=long((Y[i]-meanY[0])*(Y[i]-meanY[0]));
}
for (j=1; j<n; j++)
{
    StdX[j]=StdX[j-1] + X[N-1+j]*X[N-1+j] - X[j-1]*X[j-1] +
        N*long(meanX[j-1]*meanX[j-1]-meanX[j]*meanX[j]);
    StdY[j]=StdY[j-1] + Y[N-1+j]*Y[N-1+j] - Y[j-1]*Y[j-1] +
        N*long(meanY[j-1]*meanY[j-1]-meanY[j]*meanY[j]);
}

//Computing the cross-correlation
for (j=0; j<n; j++)
{
    Rjo[j]=rjo[j]/(sqrt(double(StdX[j]))*sqrt(double(StdY[0])));
    Rok[j]=rok[j]/(sqrt(double(StdX[0]))*sqrt(double(StdY[j])));
}

Rmax=-1E300;
for (j=0; j<n; j++)
{
    cout << "Rjo["<<j<<"]="<<Rjo[j] << " ;   Rok["<<j<<"]="<<Rok[j] << endl;
    if(Rmax < Rjo[j])
    {

```

```

        Rmax=Rjo[j];
        maxC=j;
    }
    if(Rmax < Rok[j])
    {
        Rmax=Rok[j];
        maxC=-j;
    }
}

sigma=0;
if (maxC >= 0)
{
    j=maxC;
    while ((Rjo[j] > Rjo[maxC]*sThreshold) && (j<n))
    {
        sigma++;
        j++;
    }
    j=maxC-1;
    while ((Rjo[j] > Rjo[maxC]*sThreshold) && (j>=0))
    {
        sigma++;
        j--;
    }
    if (j==--1)
    {
        j=1;
        while ((Rok[j] > Rjo[maxC]*sThreshold) && (j<n))
        {
            sigma++;
            j++;
        }
    }
}
else
{
    j=-maxC;
    while ((Rok[j] > Rok[-maxC]*sThreshold) && (j<n))
    {
        sigma++;
        j++;
    }
    j=-maxC-1;
    while ((Rok[j] > Rok[-maxC]*sThreshold) && (j>=0))
    {
        sigma++;
        j--;
    }
    if (j==--1)
    {
        j=1;
        while ((Rjo[j] > Rok[-maxC]*sThreshold) && (j<n))
        {
            sigma++;
            j++;
        }
    }
}

delete meanX, meanY, rjo, rok, StdX, StdY, Rjo, Rok;

return (short(maxC));
}
// End of MaxCorr

double IADiff::PreScan(const short X[])
{

```

```

    unsigned short i;
    double meanX;
    __int64 StdX;

    meanX=X[0];
    for (i=1; i<prescanWindow; i++)
    {
        meanX+=X[i];
    }
    meanX/=prescanWindow;

    StdX=0;
    for (i=0; i<prescanWindow; i++)
    {
        StdX+=__int64((X[i]-meanX)*(X[i]-meanX));
    }

    return(sqrt(double(StdX)));
}
// End of PreScan

double IADiff::Variance(double subVar[])
{
    double meanVar=0;
    subVar[nSubFrames]=0;      //MAX variation
    subVar[nSubFrames+1]=1e10; //MIN variation
    for(int i=0; i<nSubFrames; i++)
    {
        for(int j=i*prescanWindow; j<(i+1)*prescanWindow; j++)
            subframe[j-i*prescanWindow] = frame1[j];
        subVar[i]=PreScan(subframe);
        if(subVar[i] > subVar[nSubFrames])
            subVar[nSubFrames]=subVar[i];
        if(subVar[i] < subVar[nSubFrames+1])
            subVar[nSubFrames+1]=subVar[i];
        meanVar+=subVar[i];
    }
    return meanVar/nSubFrames;
}
// End of Variance

double IADiff::IADcompute(double& ITD, double& ILD, double& score)
{
    unsigned short highVar=0, sigma;
    short *postFrame1, *postFrame2;
    double maxC, sThreshold=0.5, coeff;

    meanVar = Variance(subVar);    // Compute mean, max and min variance
    if(meanVar < 2500)              // quit if mean variance is too low
        return(0.0);
    if(subVar[nSubFrames]/subVar[nSubFrames+1] < 2.5)      // max-to-min ratio
        return(0.0);    // quit if too little variation of variance
    if(subVar[nSubFrames]/subVar[nSubFrames+1] < 3.5)
        coeff=0.7;
    else if(subVar[nSubFrames]/subVar[nSubFrames+1] < 5.5)
        coeff=0.8;
    else
        coeff=1.0;
    for(int i=0; i<nSubFrames; i++)
        if(subVar[i] > coeff*meanVar)    // pick only frames with good variance
            highVar++;
    if(highVar==0)    // quit if no frames were picked
        return(0.0);

    postFrame1 = new short [highVar*prescanWindow];
    postFrame2 = new short [highVar*prescanWindow];

    highVar=0;

```



```

for(i=0; i<nSubFrames; i++)
{
    if(subVar[i] > coeff*meanVar)        // pick only frames with good variance
    {
        for(int j=0; j<prescanWindow; j++)
        {
            postFrame1[highVar*prescanWindow + j]=frame1[i*prescanWindow + j];
            postFrame2[highVar*prescanWindow + j]=frame2[i*prescanWindow + j];
        }
        highVar++;
    }
}

ITD=double(MaXCorr(postFrame1, postFrame2, highVar*prescanWindow-maxShift, maxShift,
                  sigma, maxC, ILD, sThreshold));
score=sqrt(double(sigma))/maxC;

delete postFrame1;
delete postFrame2;
// ret: part of data that's used
return(float(highVar)/nSubFrames);
}
// End of IADcompute

```

Appendix B

Class “LinSrch”: Linear Search

This Appendix contains the C++ declarations and definitions of the class *LinSrch*, which preforms the linear search for the nearest neighbors of the signal features in the 12-dimensional input space.

Declarations

```
/*
Class LinSrch: "Linear Search"
Input:   an array of "learned" ITD/ILDs with associated coordinates
        and single ITD/ILD sample.
Output:  the coordinates for an interpolated vector from the KNN closest
        matching ITD/ILD's in the "learned" set.
Method:  Linear search in 12-dimensional space.
*/

#ifndef LINSRCH
#define LINSRCH

#include <math.h>
#include <stdlib.h>
#include <fstream.h>

class LinSrch
{
private:

double PI;
unsigned int dimIAD, dimAngles, dimDist, dimLines;
float sqscale, alpha;
float **data;           // array to store the training samples
unsigned int KNN;       // K-th Nearest Neighbor to be considered

//      array   probed dist  position   index
void insert(float** , const float, const int, const unsigned int);

public:

    // default constructor
    LinSrch(unsigned int, unsigned int, unsigned int, char*, float, float);

    ~LinSrch(void);
```

```

//          signal control  scale alpha
void search(float*, float*, float, float);
};

#endif

```

Definitions

```

#include "LinSrch.h"

LinSrch::LinSrch(unsigned int dimI, unsigned int dimA, unsigned int dimD, char* filename, float SS, float a)
{
    // The constructor reads the samples form the file and builds an array to hold them
    fstream samples;
    unsigned int nLines, i, j;
    float dataLine[15];

    PI=3.14159265358979323846;
    KNN=7;                // K-th Nearest Neighbor to be considered
    sqscale=SS;
    alpha=a;
    dimIAD=dimI;          // dimension of IADs, i.e. how many numbers are there
    dimAngles=dimA;        // dimension of angular coordinates, i.e. from IAD to IAD+dimA-1 are angles
    dimDist=dimD;          // dimension of distance-coordinates, i.e. from dimA to IAD+dimA+dimD are distances

    samples.open(filename, ios::in);
    if(samples.fail())
        exit(-1);

    nLines=0;
    while(!samples.eof())    // count the number of lines in the file
    {
        for(i=0; i<15; i++)
            samples >> dataLine[i];
        nLines++;
    }
    nLines--;
    samples.seekg(0);        // reset the file
    samples.clear();

    data = new float *[15];    // make the array to hold the data
    for(i=0; i<15; i++)
        data[i] = new float [nLines];

    for(j=0; j<int(nLines); j++)    // read the data
    {
        for(i=0; i<15; i++)
        {
            samples >> data[i][j];
            if(i>=dimIAD && i<dimIAD+dimAngles)    // are angles!
                data[i][j]=float(int(360+data[i][j])%360);    // normalize for angles to be in {0,360}
        }
    }

    samples.close();
    dimLines=nLines;    //number of lines in training-samples file
}

LinSrch::~LinSrch(void)
{
    for(int i=0; i<15; i++)

```

```

        delete data[i];
    delete data;
}

void LinSrch::search(float* sample, float* coords, float SS, float a)
{
    float buffer, *NN[2], avgAngle, avgAngleSin, avgAngleCos, frac;
    unsigned int i,j;

    sqscale=SS;          // sqscale is the square of the adjustment scale
    alpha=a;             // alpha - how fast weight decreases

    NN[0] = new float[KNN];          // distance of the vector from the original sample vector
    NN[1] = new float[KNN];          // position in file / rank of the neighbor (i.e. NN, 1st NN, 2nd NN, etc.)
    for(i=0; i<KNN; i++)
        NN[0][i]=float(1e10);        // initialize for sort

    for(i=0; i<dimLines; i++)
    {
        buffer=float(0);
        for(j=0; j<dimIAD; j++)
        {
            if(j%2==0)
                buffer+=(data[j][i] - sample[j])*(data[j][i] - sample[j]);
            else
                buffer+=sqscale*(data[j][i] - sample[j])*(data[j][i] - sample[j]);
        }
        buffer=float(sqrt(buffer));    // distance in 15D space

        insert(NN, buffer, i, 0);      // store the i-th vector and establish rank in NN
    }

    for(j=0; j<dimAngles; j++)
    {
        avgAngleSin=float(0);
        avgAngleCos=float(0);
        for(i=0; i<KNN; i++)
        {
            avgAngleSin+=float(sin(data[j+dimIAD][int(NN[1][i])]*PI/180));
            avgAngleCos+=float(cos(data[j+dimIAD][int(NN[1][i])]*PI/180));
        }
        avgAngleSin/=KNN;
        avgAngleCos/=KNN;
        avgAngle=float(atan2(avgAngleSin,avgAngleCos)/PI*180);
        avgAngle=float(int(avgAngle+360.5)%360);
        coords[j]=float(0);
        for(i=0; i<KNN; i++)
            coords[j]+=(int(data[j+dimIAD][int(NN[1][i])]-avgAngle+540)%360) *
                float(pow(alpha,(-NN[0][i]/(NN[0][0] + float(1e-35)))));    // weights
        buffer=float(0);
        for(i=0; i<KNN; i++)
            buffer+=float(pow(alpha,(-NN[0][i]/(NN[0][0] + float(1e-35)))));    // sum of weights
        coords[j]/=buffer;
        coords[j]+=avgAngle+180;
        frac = coords[j] - int(coords[j]);
        coords[j]=float(int(coords[j]+0.5)%360) + frac;
    }

    for(j=dimAngles; j<dimAngles+dimDist; j++)
    {
        coords[j]=float(0);
        for(i=0; i<KNN; i++)
            coords[j]+=data[j+dimIAD][int(NN[1][i])] / (NN[0][i] + float(1));
        buffer=float(0);
        for(i=0; i<KNN; i++)
            buffer+=1/(NN[0][i] + float(1));
        coords[j]/=buffer;
    }
}

```

```

}

void LinSrch::insert(float* Dist[2], const float newDist, const int pos, const unsigned int rank)
{
    if(rank>=KNN)          // hit the bottom - throw it away!
        return;
    if(newDist<Dist[0][rank])    // compare the distance to the current NN and if closer...
    {
        insert(Dist, Dist[0][rank], int(Dist[1][rank]), rank+1);    // ...push the one there...
        Dist[0][rank]=newDist;          // ... and store
        Dist[1][rank]=float(pos);
    }
    else
        insert(Dist, newDist, pos, rank+1);    // otherwise insert it at the next position
}

```

Bibliography

- [1] Albert, A.A., "Solid Analytic Geometry", University of Chicago Press, Phoenix books, 1966.
- [2] Bell, R.J.T., "Coordinate Geometry of three dimensions", Macmillan, London, 1918.
- [3] Blauert, J., "Sound localization in the median plane", *Acustica*, vol.22 (1969/70).
- [4] Brandstein, M.S., "A Pitch Based Approach to Time-Delay Estimation of Reverberant Speech", *Proc. 1997 Workshop on Applications of Signal Processing to Audio and Acoustics New Paltz*, New York, October 19-22, 1997.
- [5] Brandstein, M.S., Silverman, H.F., "A Practical Methodology for Speech Source Localization With Microphone Arrays", *Computer, Speech and Language*, 11(2):91-126, April 1997.
- [6] Brandstein, M.S., Adcock, J.E., Silverman, H.F., "A Closed-Form Location Estimator for use with Room Environment Microphone Arrays", *IEEE Transactions on Speech and Audio Processing*, 5(1):45-50, January 1997.
- [7] Brandstein, M.S., Adcock, J.E., Silverman, H.F., "A Practical Time-Delay Estimator for Localizing Speech Sources with a Microphone Array", *Computer, Speech and Language*, Vol.9, p.153-169, September 1995.
- [8] Bub, U., Hunke, M., Weibel, A., "Knowing who to listen to in speech recognition: visually guided beamforming", *Proceedings of the 1995 ICASSP*, Detroit, MI.
- [9] Capel, V., "Microphones in action", Fountain Press, Argus Books Ltd., Hertfordshire, England.
- [10] Carr, H.A., "An introduction to space perception", Hafner, New York 1966.
- [11] Champagne, B., Bedard, S., Stephenne, A., "Performance of Time-Delay Estimation in the Presence of Room Reverberation", *IEEE Transactions on Speech and Audio Processing*, Vol.4, No. 2, March 1996.
- [12] Chan, Y., Hattin, R., Plant, J., "The Least Squares Estimation of Time Delay and its Use in Signal Detection", *IEEE Trans. Acoust., Speech, Speech, Signal Processing*, 1978.
- [13] Hartmann, W.M., "Localization of a source of sound in a room", *Proc. AES 8th International Conference*, 1990.
- [14] Hartmann, W.M., Rakerd, B., "Localization of Sound in Rooms IV: The Franssen Effect", *J. Acoust. Soc. Am.* 86(4), October 1989.
- [15] Hobbs, S.L., "Asymptotic statistics for location estimates of acoustic signals", *J. Acoust. Soc. Am.* 91 (3), March 1992.
- [16] Ianiello, J., "Time Delay Estimation via Cross-Correlation in the Presence of

- Large Estimation Errors", *IEEE Trans. Acoust., Speech, Speech, Signal Processing*, 1982.
- [17] Knapp, C., Carter, C., "The Generalized Correlation Method for Estimation of Time Delay", *IEEE Trans. Acoust., Speech, Speech, Signal Processing*, 1976.
 - [18] MacCabe, C.J., Furlong, D.J., "Virtual imaging capabilities of surround sound systems", *J. Audio Eng. Soc.*, Vol.42, No.1/2, 1994 Jan/Feb.
 - [19] Martin, K.M., "Estimating Azimuth and Elevation From Interaural Differences", Presented at the 1995 *IEEE Mohonk workshop on Applications of Signal Processing to Acoustics and Audio*, October 1995.
 - [20] Rabinkin, D.V. et al., "A DSP Implementation of Source Location Using Microphone Arrays", Presented at the *131st meeting of the Acoustical Society of America*, Indianapolis, Indiana, on 15 May 1996.
 - [21] Sommerville, D.M.Y., "Analytical Conics", Bell, London, 1929.
 - [22] Weng, J.J., "Cresceptron and SHOSLIF: Toward comprehensive visual learning", in S. K. Nayar and T. Poggio (eds.), *Early Visual Learning*, Oxford University Press, New York, pp. 183-214, 1996.
 - [23] Weng, J.J., Chen, S., "Incremental learning for vision-based navigation," in *Proc. International Conference on Pattern Recognition*, Vienna, Austria, vol. IV, pp. 45- 49, Aug. 1996.
 - [24] Wightman, F.L., Kistler, D.J., "The dominant role of low-frequency interaural time differences in sound localization", *J. Acoust. Soc. Am.* 91 (3), March 1992.
 - [25] Yost, W.A., Gourevitch, G., "Directional hearing", Springer-Verlag, New York 1987.