UNIVERSITY OF CALIFORNIA, SAN DIEGO

Audio Localization in the Automatic Cameraman

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science and Engineering

by

Evan Ira Ettinger

Committee in charge:

Professor Yoav Freund, Chair
Professor Serge Belongie
Professor Thomas Bewley
Professor Sanjoy Dasgupta
Professor Bhaskar Rao
Professor Lawrence Saul

2010

The dissertation of Evan Ira Ettinger is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

_____
Chair

University of California, San Diego

2010

DEDICATION

To my wife, Sandie, whose love and support I could not live without; to my parents, Cary and Karen, who taught me to always be curious and never be afraid to reach for my dreams; and to my Aunt Lois, whose creative spark continues to inspire me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, many thanks to my advisor, Yoav Freund, for lending me his guidance and sage advice throughout graduate school. His keen intuition for solving difficult problems never ceases to amaze me. Also, thanks to my committee, Serge Belongie, Thomas Bewley, Sanjoy Dasgupta, Bhaskar Rao and Lawrence Saul. I greatly appreciate all their time and helpful suggestions during their service on my dissertation committee.

I am also in debt to my fellow students Sunsern Cheamanunkul, Brian McFee, Samory Kpotufe, Daniel Hsu, Nakul Verma, Mayank Kabra, Matt Jacobsen and Shankar Shivappa. Their friendship and help will never be forgotten. I'd also like to thank my family and friends for their support and love throughout the years.

Chapter 3 is based on two works. The first is joint work with Yoav Freund entitled "Coordinate-Free Calibration of an Acoustically Driven Camera Pointing System" appearing in the proceedings of the International Conference on Distributed Smart Cameras in 2008. The dissertation author was the primary investigator and author of this work. The second is joint work with Sunsern Cheamanunkul, Matt Jacobsen, Patrick Lai and Yoav Freund titled "Detecting, Tracking and Interacting with People in a Public Space" appearing in the 11th International Conference on Multimodal Interfaces and 6th Workshop on Machine Learning for Multimodal Interaction in 2009. The dissertation author along with Sunsern Cheamanunkul were the primary investigators and authors of this paper.

Chapter 4 is based on unpublished work that is currently in submission. It is joint work with Yoav Freund. The dissertation author is the primary investigator and author of this work as well.

VITA

2004	B. S. in Mathematics *cum laude*; second major in Computer Science, Duke University
2007	M.S. in Computer Science and Engineering, University of California, San Diego
2010	Ph. D. in Computer Science and Engineering, University of California, San Diego

PUBLICATIONS

Evan Ettinger and Yoav Freund, "Particle Filtering on the Audio Localization Manifold", *In submission*, 2010.

Sunsern Cheamanunkul, Evan Ettinger, Matt Jacobsen, Patrick Lai and Yoav Freund, "Detecting, Tracking and Interacting with People in a Public Space", *ICMI-MLMI*, 2009.

Evan Ettinger and Yoav Freund, "Coordinate-Free Calibration of an Acoustically Driven Camera Pointing System", *ICDSC*, 2008. *Best Poster Award*.

Evan Ettinger, Brian McFee and Yoav Freund, "Automatic Cameraman", *NIPS Demo*, 2007.

ABSTRACT OF THE DISSERTATION

Audio Localization in the Automatic Cameraman

by

Evan Ira Ettinger

Doctor of Philosophy in Computer Science and Engineering

University of California, San Diego, 2010

Professor Yoav Freund, Chair

This dissertation studies the audio localization component of a touchless interactive display located in the CSE building at UC San Diego. The display has been named *The Automatic Cameraman* (TAC) and consists of four large television displays, a PTZ camera, and a microphone array. In this work, we propose a simple solution to the problem of accurately pointing the PTZ camera at speaking humans who are interacting with TAC.

The focus of this dissertation will be on a novel audio localization and tracking algorithm based on what we call the *coordinate-free* approach. Previous approaches to localization assume a precise known geometry for the microphone array. This is expressed through a coordinate system for the room with an exact position for each microphone element. As a result, arrays are typically built so that microphone positions can be known easily e.g. as linear or planar with fixed spacing. The coordinate-free method we propose requires no such knowledge of such a coordinate system allowing for an ad-hoc placement of microphones.

Our coordinate-free localization algorithm employs a statistical approach by learning a mapping from observed time-delays between microphone pairs directly to

a pan and tilt directive for the PTZ-camera. In addition, we explicitly utilize the fact that the training set of time-delay vectors lie on a low-dimensional structure, namely a three-dimensional structure governed by the sound source's true spatial location. We explore various regressor models with special attention to those that are known to exploit this intrinsic low dimensionality.

We follow this with a study of a particle filtering based tracker of the time-delays between microphones. Our tracker employs a novel approach to the particle filtering problem based on online learning. It introduces a new, practically useful, particle resampling scheme. It is also more robust to model misspecification than traditional particle filters.

In the final part of the dissertation, we examine a MEMS digital microphone based array that we recently implemented on an FPGA. We explore how this digital array will alleviate many of the technical deficiencies of the current analog array in TAC.

Chapter 1

Introduction

Touchless interactive displays (TIDs) have the potential to revolutionize video-conferencing, media advertising, video-game playing, consumer electronics and many other industries. Much of the interest surrounding TIDs is drawn from the promise of interaction with the device without any additional special equipment. No special gloves, remotes, mice, keyboards, touchscreens or any other input devices are required. The user is able to interact in an intuitive and natural manner through gestures and speech alone.

Consider the quintessential example of the television. One of the greatest and simplest inventions associated with modern televisions is one we often take for granted today – the remote control. The remote allows us simple access to much of the television's functionality such as channel changing or adjusting the volume and with the convenience of distance away from the screen. The remote transformed the television landscape. Viewers no longer had to watch a particular program because they didn't want to get up and change the channel. It even gave birth to a new phrase in common vernacular: "channel surfing." As a result of the shortened attention span of viewers, broadcasters changed the way they presented their programs. This reaction included more product integration into programs since commercials were watched less, the seamless transition between two adjacent programs so as to not lose the attention of viewers, and many other tricks to keep viewers on the channel. The remote control had a huge impact on the way we watch television and the way content providers broadcasted programming.

Nevertheless, the remote control is not without its critics. The most often heard complaints are that people have too many remotes, doing intricate operations from the remote such as adjusting the picture can be complicated, and that the remote itself is often hard to find when one needs it. A touchless interactive television has the promise of keeping the same convenience of the remote control, but without the need for the physical device. Imagine an interactive TV that could be controlled by gestures recognized by a camera integrated into the television. A flick of the hand up or down may indicate a channel change. More complicated operations may require a heads up display be shown on the television which the user can navigate via hand gestures.

Although this may not silence all the critics of the traditional remote control, it would without a doubt allow for a more personal interaction and a similar TV revolution like that of the remote. The hope is that novel user interface designs could be made to be more efficient at achieving the end-user's control intentions. This is not a far-fetched hope. Novel user interfaces based on more natural gestures are starting to appear in the marketplace. For example, the success of touchscreen devices such as the iPhone, iPad and its derivatives are indicators that a system based on natural gestures can become very popular. It is thought that a similar breakthrough is waiting to happen through gestures at a distance. Development of gesture based TVs has already begun and products such as Microsoft's Project Natal[1], a controller-free gaming system, show that industrial interest is peaking.

This thesis surrounds the development of a TID at UC San Diego called *The Automatic Cameraman* (TAC). TAC is a passive interactive device that senses when a user is present in front of it and then offers a device-free interactive experience to the user. In particular, the user has the option to start and stop a video recording of themself through hand gestures alone. The system has been fully operational since December of 2008 and has collected over 1500 videos. This equates to over 25 hours of video footage wherein people are actively interacting with TAC.

Specifically, this dissertation surrounds the techniques we use in TAC to localize audio sources with a microphone array. We will discuss the general methodology behind audio localization at the end of this chapter, but first to put us in the proper context we

[1]More information can be found at `http://www.xbox.com/en-us/kinect`

Figure 1.1: Frontal view of TAC display unit. PTZ camera and four of the seven total microphone are visible.

describe TAC and its history in more detail.

1.1 The Automatic Cameraman (TAC): Hardware

TAC is a touchless interactive display located on the 4th floor lobby of the Computer Science and Engineering building (EBU-3b) at UC San Diego. The display consists of four 52” plasma TVs placed in a 2x2 grid on a wall. The unit is prominently displayed to public traffic coming through the lobby either by elevator or via the main stairwell. A frontal view of TAC can be seen in Figure 1.1. There are two main sensing modalities that TAC uses: a pan-tilt-zoom (PTZ) video camera and an array of seven microphones. The PTZ camera is mounted near the top-middle of the display and it can be controlled via a local network connection. The microphone array has microphone elements at each corner of the display and the remaining three placed in a triangular pattern on the ceiling. This arrangement of sensors is shown in Figure 1.2.

Figure 1.2 also depicts the dimensions of the lobby. The lobby is rather large (approx. 13m x 10m x 6m). There are a pair of elevators directly across from the

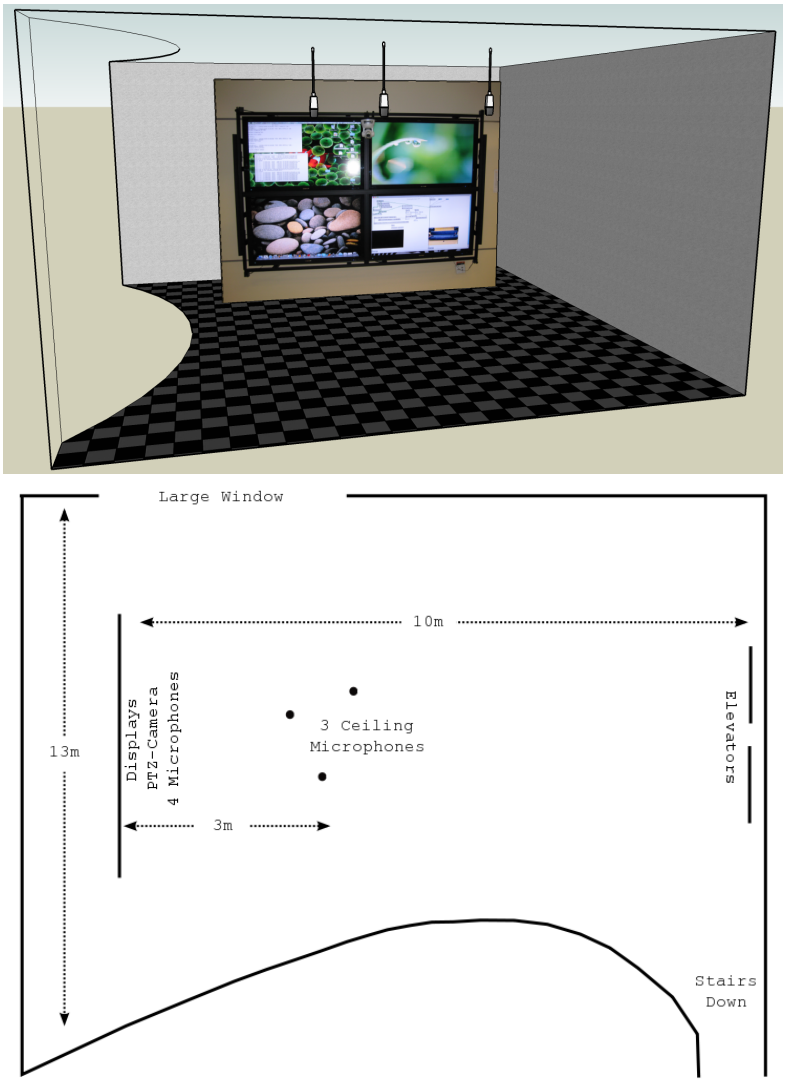


Figure 1.2: Top: sketch of the 4th floor lobby and placement of sensors. Bottom: Birds eye view of room layout with labeled dimensions.

display and a large floor-to-ceiling window to the left of the display that changes the lighting in the room significantly depending on the position of the sun and on weather conditions.

The two sensing modalities utilized by TAC are video via the PTZ camera and audio from the microphone array. Nearly all signal processing is done on a 2.66 GHz quad core G4 Mac located on the opposite side of the wall from the displays. In addition,

a portion of the signal processing is performed on an FPGA and we discuss this in further detail in Chapter 5.

The microphone array is used to discover the location and then track any audio sources present in the lobby. In addition, it is also used to perform audio enhancement to the source by combining all microphone channels into one that is steered to the source. We use a beamforming technique that focuses on the sound source of interest and attenuates any noises coming from other locations. We discuss this further in Chapter 4.

The video stream from the PTZ camera is also processed by the Mac desktop computer. TAC runs a Viola and Jones style face detector [VJ01] trained via AdaBoost on examples that have been collected by TAC since it was deployed in late 2008. After a face has been detected, we then utilize a light-weight face tracking framework that consumes much fewer computationally resources than running the original detector on subsequent available frames. In addition, a simple color-based skin detector is implemented that outputs whether a given pixel is likely to be part of a human skin region.

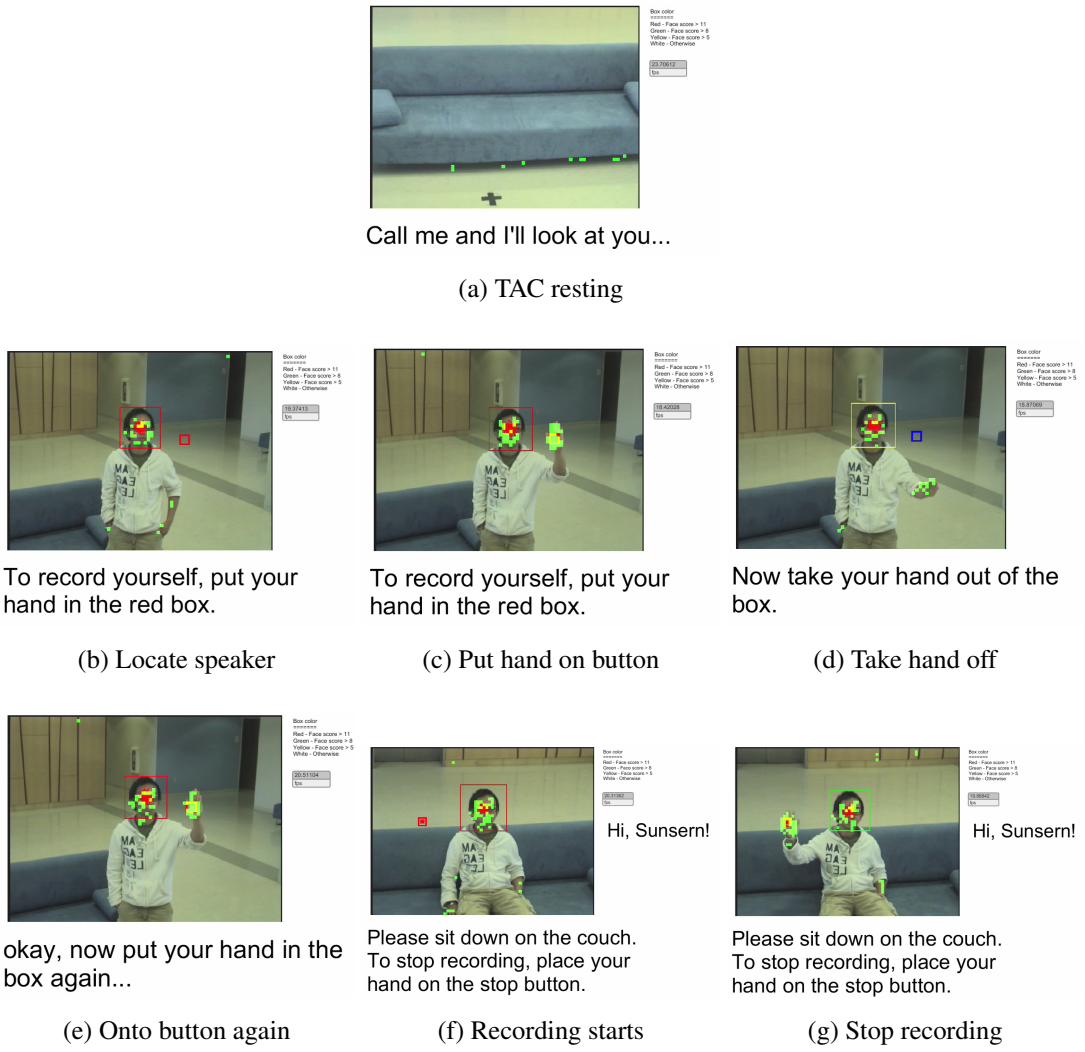
The audio localization, face detection and skin detection are the only way in which TAC can sense its surroundings. It utilizes all of this information in concert in order to make a fully interactive experience for the end-user. In the next section, we describe step-by-step how a typical user-interaction with TAC plays out.

1.2 The Automatic Cameraman (TAC): User Interface

Often users begin to anthropormorphize TAC after interacting with it for a short while. Users refer to TAC "looking" at places as if it were alive and the PTZ camera were its single eye. We will refer to it in a similar manner in the description of the user interaction protocol below, that is we will use terms such as "looking" to mean the camera is pointing at the user, or "turning" to mean the camera moves to point at the user, etc.

TAC gives users the ability to make their own video recordings of themselves. The protocol for recording a movie with TAC is depicted in Figure 1.3. The protocol is as follows:

(a) Resting: When a user first approaches TAC, it is in its resting state looking down at



(a) TAC resting

(b) Locate speaker

(c) Put hand on button

(d) Take hand off

(e) Onto button again

(f) Recording starts

(g) Stop recording

Figure 1.3: The protocol for getting TAC's attention and starting/stopping recordings.

the floor away from the people that frequently pass through the lobby.

(b) Locating the user: To attract TAC's attention, the user must make a sound like "Hey look at me". The sound must be significantly loud and sustained for a brief period of time. That is, it can neither be too quiet nor too brief like that of a single clap. The sound is then located via the microphone array and the PTZ camera is directed to point at the calculated location. Then, the face detector finds the user's frontal facing face and centers it in the field of view.

(c)-(f) Starting a recording: An on-screen button is presented to the user in the form

of a small red box. To start a recording the user is first prompted to place their hand over the box for a few seconds. The system then asks them to take their hand out and then back inside the box to ensure that they would like to start a recording. This button protocol reduces the number of videos that start up without any user-intent. At this point a recording of the user begins via Quicktime.

(g) Stopping the recording: When the recording starts, a stop button is immediately displayed to the user on the opposite side of the screen from where the start button was. The recording stops when either the user places their hand over this button or when a face has not been detected for 15 seconds (i.e. the user has left the field of view without stopping the recording).

In addition, TAC can also recognize who the user is if they have been registered as a popular user of TAC. In Figure 1.3 TAC has recognized Sunsern Cheamanunkul, one of the primary developers associated with TAC.

The virtual button is activated by placing a skin region over it, typically a hand. If the number of detected skin pixels inside the button is above a threshold consistently across a duration of a few seconds, then the button is considered pressed. This button is greatly inspired by the work of William Freeman and collaborators [FR95, FW95, FBK$^+$99] who show that simple computer vision algorithms can be very effective for HCI applications if the feedback from the system is fast. If the reaction by the system is fast to user input, typically under 100ms, the user's hand-eye coordination is engaged allowing for the interaction to feel natural [MMRC92, Mil68].

This protocol for recording videos has proven very useful for collecting interesting data to learn about the users of TAC. Each video recording is performed at 640x480 and then compressed using H.264 compression technology. For the video's audio track we use the beamformed audio track that is combined from all 7 microphone channels. As of the writing of this thesis, since December 2008, when TAC first became live, over 1500 videos have been collected. This equates to over 25 hours of footage of people interacting with TAC[2]. Later on we also started collecting all 7 channels of microphone audio throughout the duration of the user's recording. Several hours of 7 channel audio has also been collected in WAV format.

[2]Videos can be browsed at `http://seed.ucsd.edu/~cameraman/video_recordings`

These recordings are a fruitful source of data on which to improve TAC. In fact, the philosophy behind TAC's development has been proposed as a feedback loop. By having a repository of user's interacting with TAC we can learn and develop interesting new ways for TAC to sense it's users and environment. In doing so, richer and more intricate applications can be developed allowing for even longer and more interesting user interactions to occur. In Chapter 3 we show how leveraging this data over time can improve TAC's ability to localize accurately, an indication that such an approach to improving TAC has great potential.

1.3 The Automatic Cameraman (TAC): Software Design

In this section we discuss the software design of TAC summarized in Figure 1.4. The input signals into TAC are both video and audio streams. The video is captured by a Sony SNC-RZ30N network PTZ camera at 640x480 resolution at 30 fps, and then digitized by a capture card in the main computer. The audio input consists of 7 CAD CM100 condenser microphones connected by 15' XLR cables to a MOTU 896HD digitizer. The digitizer samples each channel at 24-bit precision and a sampling rate of 16 kHz, which is high enough for the full range of human voice to be well below the Nyquist frequency. The audio digitizer is connected to the main computer via a firewire connection.

The software design of TAC is derived from the model-view-controller paradigm. User inputs come in the form of either talking or making gestures and appearances in the field of view of the camera. These inputs must then be translated via the model into appropriate intentions. This translation is two-fold. First, signal processing modules attempt to extract relevant information from these modalities such as audio or face locations. Second, the result of these signal processors are sent to a central unit that maintains both a combined model of recent activity and translates this model into user-intentions in the form of control commands to other external entities. This centralized unit could easily be called the brains of TAC and is labeled *controller* in Figure 1.4. It is responsible for combining the information from multiple signal processors and reacting with appropriate control commands.

The three events received as input to the controller are audio location beliefs

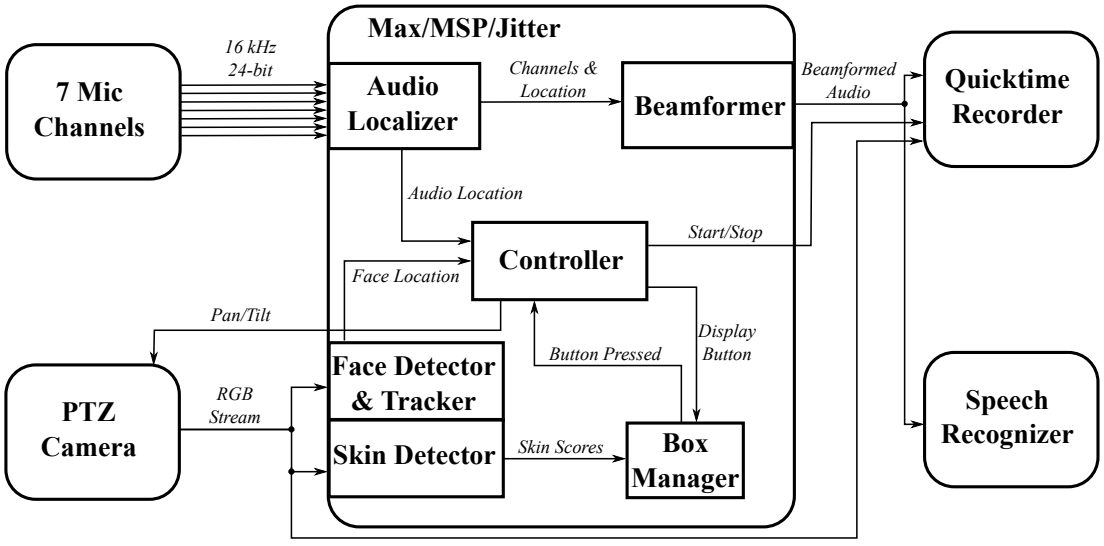


Figure 1.4: A flow diagram of the software components involved with TAC.

from the *audio localizer*, face location beliefs from the *face detector*, and button press events from the *box manager*. These units along with the *beamformer* and *controller* are implemented as functional units in the Max 5 graphical programming environment[3]. Max allows for quick and simple development of signal processing applications by a simple graphical programming model. Programming at a high level in Max involves connecting functional units called *patches* to each other via edges that carry signal data. For example, the input to the audio localizer would be the audio signals and the output a signal indicating location beliefs in the form of pan and tilt directives for the camera. Although these functional units can be connected simply through this graphical programming language, the underlying functional logic of each is implemented using traditional programming languages, namely C or Java.

The precision and accuracy of the signal processing components are essential for the controller to even have a chance of finding and centering the user's face in the field of view. The audio localizer utilizes the PortAudio C library[4] to gain access to the 7 digitized audio channels. It maintains a window of the latest 500ms of audio for each channel updated every 25ms. On each update of the window it first decides whether there was a significant noise made in the room that is above a certain energy threshold

[3] Available at `http://www.cycling74.com`
[4] Available at `http://www.portaudio.com`

in the audio signals. If so, then the localization scheme discussed in Chapters 3 and 4 is performed producing a pan and tilt directive for the camera. The directive is a belief on where to point the PTZ camera so as to center the producer of the noise in the field of view. This directive can be given to the beamformer which creates a single channel of audio focused on the location of the speaker. The beamformer implements a delay-and-sum methodology.

The face detector is implemented as a Viola and Jones style face detector [VJ01] trained on hand-labeled examples of faces collected by TAC over its existence. It utilizes a feature set based on simple edge detectors and scores from the pixel-based skin detector. The visual processing components are not the focus of this thesis, but further details of it can be found in [CEJ+09]. The tracking methodology utilizes a particle filtering scheme that is very similar to what is discussed in Chapter 4. The face detector outputs face location beliefs in the form of a pixel-position (x, y) of the center of the face and the size of the detection box.

The box manager handles the display of the on-screen button and takes as input the relevant skin-pixel scores from the skin detector. If the average score inside the box is above a threshold for a duration of 3 seconds, then the button is considered pressed and it relays this information to the controller.

Each of the outputs of the signal processing units are a noisy stream of events going into the controller. For example, the face detector stream is noisy since it does sometimes detect non-faces and give an incorrect location belief. The same holds for the audio localizer. The controller must manage and filter these streams to extract whether the user is in fact present and in the center of the field of view. If the controller determines that the user is not properly centered, then the controller must take action to correct this.

The strategy employed by the controller is a simple policy combined with median filtering over the localization events from the recent past. To formalize what we mean by recent past, let E be the set of events communicated to the controller in the past N seconds, where N is a parameter that needs to be tuned. To determine whether the camera moves to look at a new audio source, there must be at least k audio events in E clustered near the median of E. If found, the camera is directed to center on this median

location. A similar set of simple control logic is used for fine adjustments to center the user's face in the field of view.

The camera begins looking for a face after a sound has been heard and the camera directed to point to the calculated location. A similar median filter of E is performed before directing the camera to make slight adjustments for centering the face and zooming so that it is of the appropriate size. To avoid constant small adjustments, PTZ changes to the camera are only made if the position or size is not within an acceptable range where the face is near centered.

The controller constantly monitors E determining whether a new sound source should be looked at, and whether the user's face location or size calls for PTZ corrections. It communicates these PTZ directives via a local network using the Sony designed VISCA camera control protocol. The controller is also responsible for starting and stopping video recordings with Quicktime and then sending the recorded videos to a webserver for external viewing. Communication with Quicktime is performed through Apple's ActionScript, and other scripts handle the delivery of the recordings to the webserver. The beamformed audio is sent to Quicktime via the inter-application audio routing utility Soundflower[5].

In addition, the controller records an annotated log of audio and face detection events that happened throughout the user's video recording. The annotations contain such things as face detection score values and more detailed audio location beliefs. Not only is this useful for debugging purposes, but also serves as a means to collect other interesting quantitative data for improving both the face detector and audio localizer. We discuss these logs in further detail later on in the thesis since they are used for collecting data on which statistical models are trained on.

1.4 Audio Localization in TAC

This thesis surrounds the design and implementation of TAC, a touchless interactive display. In particular, the focus of this thesis is on the methodology behind the audio localization technology in TAC. In this section we introduce the fundamentals of

[5] Available at `http://cycling74.com/products/soundflower`

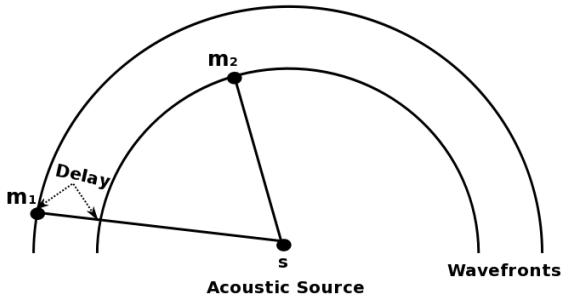


Figure 1.5: The delay associated with microphones m_1 and m_2 caused by the spherical waves of propagation of the sound source s.

audio localization from a high-level and also discuss the novel approach we design for it.

First we must discuss the physical principles governing audio localization, or what is otherwise known as *passive sonar*. This technique is passive since the localization technology is only *listening* for detectable noises to then locate. This is opposed to active sonar which searches for objects by *emitting* an ultrasonic sound in a particular direction and then measuring echoes.

To start, we introduce the underlying principles behind passive sonar. Let's assume we have access to a pair of spatially separated microphones. Imagine a brief point sound like the pop of a balloon or a quick clap of the hands originating from a fixed location in space. A good first order approximation for the movement of sound in air is a spherical propagation centered at the sound source. The propagating waves expand out with time at the speed of sound. This is depicted for a 2-d model of the world in Figure 1.5. The result is a delay in arrival of the instantaneous sound at each microphone. This delay is known as the *time delay of arrival* or TDOA for short.

The task of audio localization is to take several microphones, first estimate the TDOA between all pairs of microphones, then transform these estimates into a location belief for the sound source. The problem at hand is similar, but more difficult than that of trilateration. In trilateration one first measures the distances to the object of interest from several beacons whose locations are known a priori. By finding the point of intersection between the resulting spheres centered at each beacon one can calculate the location of the object.

The task of audio localization can be posed in a similar fashion except only relative differences of distances (or times of arrival) are known between pairs of beacons. This task is called multilateration or hyperbolic positioning. The latter name results from the fact that a fixed TDOA between a pair of beacons with known positions causes the positions for the source to lie on a hyperboloid of one sheet, instead of on the surface of a sphere like in trilateration. Similarly, when multiple TDOAs are known for many beacons these hyperboloids of one sheet can be intersected to calculate the feasible locus for the source.

Multilateration has proven useful for locating an object that emits a source to several beacons, such as an aircraft emitting a distress signal. It can also be used for an object to determine its own location if the beacons are synchronized to emit a signal at the same time towards the receiving object. This latter application of multilateration is useful for navigation.

One difficulty in this approach to multilateration is that often the estimates for the TDOA are noisy, especially in the case of human speech. When speech is examined at time scales of milliseconds, the signal is often very periodic with a similar repeating pattern making TDOAs difficult to estimate accurately. This is especially true for voiced speech elements more so than unvoiced since the former is formed by the vibrations of the vocal cords. Given that some of the TDOA measurements may be noisy, it is often the case that the intersection of resulting hyperboloids is empty. Instead, a non-linear optimization problem is often solved, for example using a least squares technique, to find the point in space closest to all hyperboloids. In addition, a tandem tracking methodology over measurements across time proves useful for stability, for example a simple averaging or a Kalman filter.

Another difficulty arises from the need to know position estimates for the beacons (microphones). For TAC the lobby in which people are to be localized is quite large. People may be far from the display and we still want the PTZ camera to be able to accurately point at them. Small errors in the position of each beacon can become amplified when calculating the source's location. This is especially true with increasing distance of the source from the beacons. For example, consider a pair of microphones and a far away object. A good first order approximation of what the TDOA tells us is

simply the bearing of the object relative to the line between the microphones. With errors in the microphone positions this bearing becomes inaccurate. Subsequently, errors on the orders of centimeters quickly become meters with increasing distance.

To be able to get good coverage in the lobby, we are required to put the microphones relatively far apart – on the orders of several meters apart. This makes accurately developing a coordinate system in which microphone elements are placed difficult to do by hand. Moreover, this coordinate system must be aligned with the PTZ camera as well. Since we are only concerned with pointing the PTZ-camera to the speaker, knowing an absolute position for the emitting sound source is not necessary. TAC only requires the appropriate PTZ commands to center the source in the field of view.

Therefore, we need a novel approach to translate delays into PTZ commands. The traditional approach would introduce a careful calibration and then position estimation technique. We eliminate this intermediate step and propose a "coordinate-free" approach. We do not require any knowledge of a coordinate system wherein microphone locations are known. This allows for a much simpler approach to be taken described in Chapter 3 based on regression and statistical modeling. An additional important benefit of this approach is a passive continuous calibration of the audio localization system. This is performed by collaborating with the face detector and the controller to collect relevant new data to refine the statistical models over time.

The method we introduce in Chapter 3 does not make any temporal assumptions in the statistical model for making PTZ predictions. The method presented there solely focuses on making accurate predictions for the appropriate PTZ given the most recent frame of audio from each of the microphones. There is no temporal information taken into account such as predictions from previous frames.

In Chapter 4 we introduce a novel tracking framework that does take temporal information into account. The tracker makes the implicit assumption that people do not move too quickly (e.g. jump around the room instantaneously). As a result when we are confident we have located a person, the estimates for PTZ commands should not change much in subsequent frames. Our methodology is a variation on a particle filter but exploits various domain specific knowledge about the localization problem. In addition, the tracker can still make accurate predictions when individual TDOA estimates fail or

are of poor quality. This allows TAC to explicitly leverage the redundancy available to it from the 7 element microphone array (a minimum of 4 microphones are needed given perfect TDOA estimates).

No calibration of the array is needed since the localization unit can update itself through feedback collected from the face detecting unit. Collecting an initial set of data is recommended for making the unit quickly functional, although not required. Luckily collecting such an initial calibration set is as simple as talking and interacting with the system at various points throughout the area of localization interest. The final result is an audio localization unit that gives accurate estimates of a constant noise source's location every 25ms.

1.5 Thesis Organization

Given that TAC has 7 microphone elements in its array, there are $\binom{7}{2} = 21$ unique pairs of microphones for which a TDOA estimate can be calculated. If we concatenate these TDOAs together we can represent the ensemble in a vector space in $\mathbf{R}^{21}$. Each physical location for a sound source corresponds to a 21-D vector in this space. However, most locations in this vector space do not correspond to a sequence of TDOAs that are feasibly created by any sound source's 3-d location. This is because there are only three degrees of freedom of this system, namely the three spatial dimensions in which a sound source can vary. Furthermore, small variations in the sound source location leads to small variations in the resulting 21-D TDOA vector. In fact, in a very real sense that we formalize in Chapter 2 the surface on which these TDOA vectors may lie on is of dimension exactly 3.

Generating a statistical model of this surface, called the TDOA manifold, is one of the central contributions of this thesis. In Chapter 2 we discuss the tree-based models of the structure of this manifold. This model of the manifold becomes central for both the audio localization and tracking methodology that is developed in Chapters 3 and 4 respectively.

One of the central problems encountered while building TAC was the poor quality of audio captured by the analog microphones. This issue stems from the fact that

analog signals are highly sensitive to RF/EM interference. For each microphone we require a 15' cable to attach the microphone to the audio digitizer. The cables, despite being shielded, run near multiple sources of electrical interference and are exposed to many wireless signals present in the UC San Diego Computer Science building. The result is a poor signal to noise ratio, that is far from ideal. Chapter 5 discusses our proposed solution to this problem through the use of MEMS digital microphones and an FPGA. The MEMS microphones are extremely small microphones etched into silicon (3mm wide). More importantly for improving interference from other signals the MEMS have an analog-to-digital converter built in as part of the microphone meaning the microphone transmits a digital signal, a very robust solution to signal interference. This makes MEMS microphone optimal for use in laptop computers, cell phones and other devices where signal interference due to close-by electrical componenets is inevitable.

These microphones do not have standard audio connectors as of yet. We resolve this issue by connecting them to the pins of an FPGA which we program to decode the digital signals into traditional audio signals. The details surrounding this FPGA implementation is discussed in Chapter 5. Finally, some directions for what the future of TAC should be is discussed in Chapter 6. In particular, we discuss the applications that are currently being worked on by other graduate students and those that are of interest in the near-future.

Chapter 2

Tree-based Manifold Models

In this chapter we discuss the tree-based models of the feasible TDOA region, a central component to the statistical models used for audio localization and tracking. The models presented here are space-partitioning algorithms that build a tree hierarchy with leaves representing a cell of the partition wherein a simple (e.g. linear) model can be stored. When these simple models are combined as a whole across all leaves, the result is a robust method of developing piecewise models of a manifold.

The method is robust in the sense that it can provably adapt to a specific type of low-dimensional structure present in the data e.g. if the data is sparse or lies near a low-dimensional structure. The latter is of interest to modeling the TDOA manifold: despite the fact that in TAC a series of TDOA measurements is of dimension 21, we still know that there are only three underlying degrees of spatial-freedom in which a sound source can vary to generate such a series of TDOAs between microphone pairs. We now describe an efficiently learnable class of tree-based models, random projection trees, that can exploit any present low dimensional structure in the dataset.

It is worth noting that the random projection tree algorithm is due to Dasgupta and Freund, and that the next section is describing work that can be found in [DF08]. This is not work performed by the author of this thesis. The fundamental ideas presented in the next section are central to the work that follows in the remainder of this Chapter and the next two.

2.1 Random Projection Trees

Algorithm 1 Recursion for building a kd-tree

Require: A dataset $S \subset \mathbb{R}^D$.

1: **Projection:** Choose a coordinate axis $i \in \{1, \ldots, D\}$. Let $m = median_{x \in S}(x_i)$

2: **Assignment:** Let $S_L = \{x \mid x \in S \text{ and } x_i \leq m\}$ and $S_R = S \setminus S_L$.

3: **Recurse:** Recurse on S_L and S_R.

The random projection tree (RP-Tree), a space partitioning tree first analyzed by Dasgupta and Freund [DF08], is a randomized algorithm constructed in a very similar manner to the popular kd-tree. Like its kd-tree relative, the RP-tree can be used for nearest neighbor retrieval, vector quantization, regression or classification. Recall that a kd-tree is built by recursive binary splits of the dataset where each split is chosen to be the median of a single coordinate direction. RP-Trees are also built by recursive binary splits, but instead of a coordinate direction, the data is first projected onto a random direction and then split near the median. Pseudocode for both the kd-tree and RP-Tree are given in Algorithm 1 and Algorithm 2 respectively.

Note that the RP-Tree algorithm presented here is a slightly simplified version of the one presented in [DF08]. To achieve the theoretical results presented there, first they add a small amount of randomness to the median value for the splitting point. They also consider another type of split based on the distance from the mean of the projected values. Nevertheless, since in this thesis we consider only experimental results with the RP-Tree and its relatives, this simplified version will suit our needs.

If a low-dimensional structure is present in the dataset, then the RP-tree can discover and exploit it much quicker than its kd-tree counterpart. We state more precisely this difference in adaptability in what follows. To start, we describe the notion of *intrinsic dimensionality* of a general set $S \subset \mathbb{R}^D$. Despite the fact that S lies in an ambient dimension of size D, the set may have a regular structure that allows for it to be of much lower intrinsic dimensionality $d << D$. The theoretic work of Dasgupta and Freund has shown that the RP-tree can adapt to two very specific notions of intrinsic dimension: Assouad dimension and local covariance dimension.

The Assouad (or doubling) dimension is a popular way of measuring intrinsic

Algorithm 2 Recursion for building an RP-Tree

Require: A dataset $S \subset \mathbb{R}^D$.

1: **Projection:** Choose a random unit direction $u \in \mathbb{R}^D$. Let $m = median_{x \in S}(x^\top u)$

2: **Assignment:** Let $S_L = \{x \mid x \in S \text{ and } x^\top u \leq m\}$ and $S_R = S \setminus S_L$.

3: **Recurse:** Recurse on S_L and S_R.

dimensionality that often coincides with our intuition of what any definition ought to include. More precisely, let $B_r(x)$ be a ball of radius r centered at $x \in \mathbb{R}^D$. A dataset S has Assouad dimension d if for any x, $B_r(x) \cap S$ can be covered by 2^d balls of radius $r/2$. This definition of intrinsic dimension includes d-sparse datasets and a wide class of d-dimensional Riemannian manifolds [DF08]. For example, consider a straight line segment in $\mathbb{R}^D$: any $B_r(x)$ centered on x on the segment can be covered by exactly 2 balls of $r/2$ giving an Assouad dimension of 1.

However, we are often interested in only samples of a dataset that we believe are drawn from an underlying distribution that lies near a set S of low intrinsic dimension. It is difficult to verify that S has low Assouad dimension from just a finite sample. It is worth stressing that in practice we only believe that data lies "near" S since noise is often observed. Therefore, a more practical way of measuring the intrinsic dimensionality of a dataset is via the *local covariance dimension*, which we now describe. This notion of dimensionality formalizes the intuition of how well a linear subspace can describe a dataset at a particular bandwidth. More specifically, a set S has local covariance dimension (r, ε, d) if all balls of size r have at least $(1-\varepsilon)$ fraction of the eigenspectrum of the covariance matrix contained in the top d eigenvalues. That is,

$$\sum_{i=1}^{d} \lambda_i \geq (1-\varepsilon) \sum_{i=1}^{D} \lambda_i \tag{2.1}$$

where λ_i are the eigenvalues (ordered from largest to smallest) of the covariance matrix for the ball of size r. This definition coincides with the intuitive notion that at small scales a Riemannian manifold is well described by a hyperplane. The same intuition has been used in various manifold learning algorithms [RS00]. At the appropriate scale, the dataset is locally near-linear. Moreover, this definition is easily empirically verified by examining the covariance matrix of a sample drawn from the underlying distribution.

Regardless of whether Assouad or covariance dimension is used as the specific notion of dimensionality, the guarantee given in [DF08] is that given a data sample that has low intrinsic dimensionality the tree will quickly halve the data diameters in cells as you descend it. More specifically, if the data in a particular cell C has Assouad dimension d, then with constant probability the RP-Tree will halve the data diameter at a descendant cell roughly $d \log d$ levels below. A similar result can be shown for an RP-Tree using the local covariance dimension instead. Decreasing the data diameter quickly is of interest for tasks such as vector quantization, regression and classification. For example, in vector quantization this means that the depth needed to achieve a particular quantization error is much less than the depth needed by the kd-tree counterpart – in other words, many less quantizers are needed if the intrinsic dimension is low. The kd-tree, in the worst case would need D levels to halve the data diameter. The example that achieves this worst-case complexity is a dataset in $\mathbb{R}^D$ that is concentrated exactly on the coordinate axes. Despite this dataset being sparse, and having small Assouad dimension ($\log 2D$), the kd-tree only splits along coordinate axes causing it to take D splits to halve the diameter.

An RP-Tree is therefore ideal for modeling the TDOA manifold, the main object of interest in the statistical models of Chapter 3 and 4. In TAC, each set of TDOAs between all pairs of microphones lies in $\mathbb{R}^{21}$, however, the underlying process in which these TDOAs are created has only three degrees of freedom: the spatial degrees in which a physical sound source can vary its location in. As the TDOA data has low intrinsic dimension, the RP-Tree will partition that data into regions of small diameter in a small number of levels. In other words, the leaf cells of the binary partitioning should quickly converge to contain data which lies close to a 3d affine space.

2.2 Principal Direction Trees

As described in the previous section, RP-Trees are able to efficiently reduce the diameter of the dataset as the depth of the tree increases. Surprisingly, this result is achieved despite the RP-Tree being a passive algorithm that can choose the random directions independently from any data sample. That is, the RP-Tree doesn't learn any

Algorithm 3 Recursion for building a PD-Tree

Require: A dataset $S \subset \mathbb{R}^D$.

1: **Projection:** Let $u \in \mathbb{R}^D$ be the top principal direction of S and be of unit length. Let $m = median_{x \in S}(x^\top u)$

2: **Assignment:** Let $S_L = \{x \mid x \in S \text{ and } x^\top u \leq m\}$ and $S_R = S \setminus S_L$.

3: **Recurse:** Recurse on S_L and S_R.

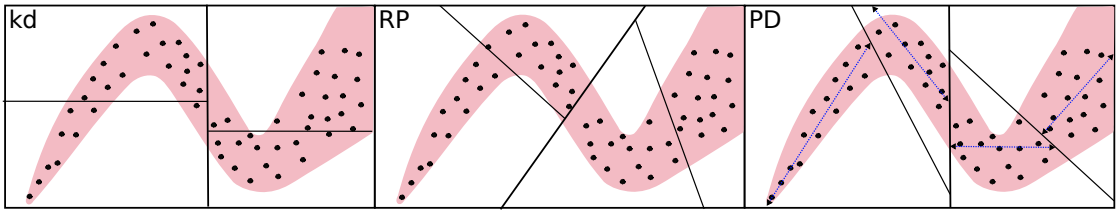


Figure 2.1: A toy dataset whose distribution is shown in pink with depth 2 partitioning trees (left: kd-tree, middle: RP-Tree, right: PD-Tree). Blue vectors indicate principal directions on the PD-Tree.

properties of the data in order to choose directions on which to project on[1]. It is nevertheless able to converge to local regions of the dataset quickly.

In practice, we are often able to achieve in even faster reduction of diameter by examining a simple property of the dataset: its top principal direction of variance. By calculating the largest direction of variance and splitting the dataset at the median, it is intuitive that a reduction in diameter will happen at a rate proportional to the intrinsic dimension d. Moreover, in situations where the top direction of variance is much larger than any other orthogonal direction we can achieve very fast reductions. We call this type of a tree a principal direction tree (PD-Tree), with pseudocode for its construction given in Algorithm 3. Empirical results on various datasets have show that a PD-Tree converges to local regions at a significantly faster rate than an RP-Tree (and kd-tree) [VKD09]. The PD-Tree will be used predominantly in the experimental work to follow, directly inspired from the theoretical results surrounding the RP-Tree discussed in the previous section. A comparison of a kd-tree, an RP-Tree and a PD-Tree on a toy dataset are given in Figure 2.1.

[1]however, note that it does need to learn the medians of the projection values

The additional cost in terms of computational efficiency in building a PD-Tree is not large compared to that of an RP-Tree. For an RP-Tree, to create two children nodes we must compute the projections of all N D-dimensional datapoints contained in the parent node at a computational cost of $\mathbb{O}(ND)$. The only additional cost of a PD-Tree is that of computing the top principal component. A naive implementation would simply do an entire principal components analysis to extract the top direction. However, this also computes the unneeded lower eigenvectors as well.

In practice, much simpler methods exist to compute the top principal direction of a dataset. For very large datasets where a PCA is impractical, we use a simple online update rule first presented in [WZsH03]. Assuming the data has mean zero, initially, the top direction is set to be equal to the first example, $v(0) = x_0$. The update equation upon seeing each subsequent example is as follows,

$$v(n) = \frac{n-1}{n} v(n-1) + \frac{1}{n} x_n x_n^\top \frac{v(n-1)}{\|v(n-1)\|} \tag{2.2}$$

This update is derived from the observation that the top principal direction obeys the eigenvector equation $v = \lambda u = \Sigma u$, where Σ is the covariance matrix of the data and u is its unit-length eigenvector that corresponds to the top eigenvalue λ. This simple equation can be rewritten in terms of the empirical covariance matrix as,

$$v = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^\top u \tag{2.3}$$

where N is the total number of points in the data set. Thus, the update equation is an approximation of the above with u replaced by the current normalized estimate of the top eigenvector. Each update has complexity $\mathbb{O}(D)$ and the top component is typically converged on quickly making the overall complexity of the resulting PD-Tree algorithm comparable to that of the RP-Tree.

2.3 Case Example: Rotating Teapot

In this section we consider a toy experiment involving images of a rotating teapot depicted in Figure 2.2. This dataset consists of 400 images of a teapot as it does a full rotation of 360 degrees. Each image x_i is grayscale and has dimension 101 x 76. We

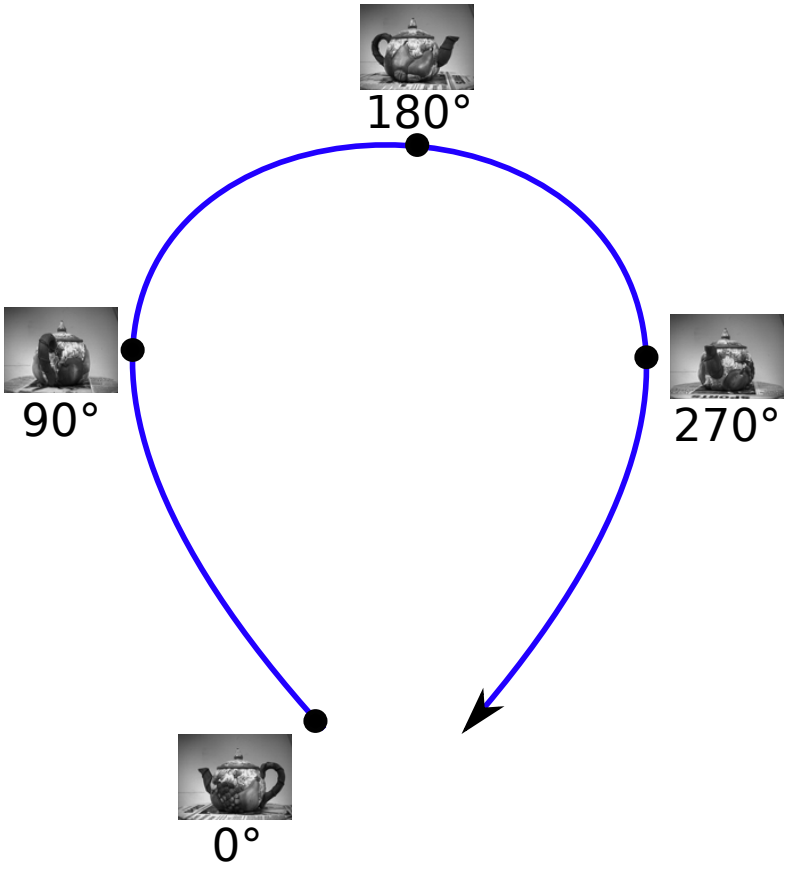


Figure 2.2: Visualization of the teapot dataset with various rotations displayed.

represent each example in pixel space, a rather high dimensional (7676 dimensional) ambient space where each $x_i \in \mathbb{R}^{7676}$. However, there is only one degree of freedom present in this dataset, namely, the amount of rotation of the teapot. Therefore, we should expect that the intrinsic dimension of the underlying manifold is also one.

However, the underlying manifold is certainly not linear. We use the online update algorithm discussed in the section above to find the top principal direction of the entire dataset. We present the projection of the dataset onto this direction in Figure 2.3. It's clear that the subspace is not linear, moreover this top principal direction does not capture much of the variance present in the dataset.

More specifically, consider the fraction of variance unexplained, F, given by the following

$$F(v,S) = \frac{\sum_{x \in S} \|x - (\bar{x} + vv^\top x)\|}{\sum_{x \in S} \|x - \bar{x}\|^2} \tag{2.4}$$

where S is the data set, $\bar{x}$ is the sample mean, and v is a principal direction. It is simple to show that $0 \leq F \leq 1$ and its interpretation is when F is small, the projection onto the subspace spanned by v explains S well. The top principal direction, v, of the entire teapot dataset has $F(v,S) = .84$, meaning it is a poor low dimensional representation of S.

We attempt to uncover a 1-dimensional embedding of this dataset where the embedding value can easily be related to the rotation of the teapot. We aim for a simple algorithm that allows for us to "unravel" the manifold via a PD-Tree into a line. First, given a PD-Tree grown to a fix depth, we would like to find nodes in the tree that have small F value and thus have good locally linear models of their respective subsets of S. Now, given subsets of S that have good linear models, how do we merge them together to form a global embedding of S?

One could easily formulate the problem by a simple calculation of nearness between nodes, for example, with Euclidean distance between the means. Nevertheless, there is a fundamental issue that must be solved with this approach: how to orient the 1-d embedding from each node relative to each other. Since we would like to unravel this manifold, we must connect disjoint nodes in the appropriate arrangement with respect to each other. In the case of the teapot dataset, we would like the rotation ordering of the examples to be preserved in a purely unsupervised manner. Distance measurements in high dimensions are a very unreliable way of deciding the orientation of these manifold pieces in relation to each other.

Instead we propose a simple alteration to the PD-Tree that creates leaf nodes which are not a perfect partition of S, but instead have datapoints in common with many other nodes. If two nodes have at least 2 points in common, then we can easily orient each projection so that the ordering of the common points in each node are identical. Next, we can connect the projections together so that these common points, now in the correct order, are as near to each other as possible. This is exactly the approach we take.

To create an overlap between internal nodes of the PD-Tree, we alter the splitting procedure so that there are two thresholds $t_1 < t_2$ that define the two children nodes as follows

$$S_L = \{x \mid x \in S \text{ and } x^\top v \leq t_2\} \text{ and } S_R = \{x \mid x \in S \text{ and } x^\top v \geq t_1\} \tag{2.5}$$

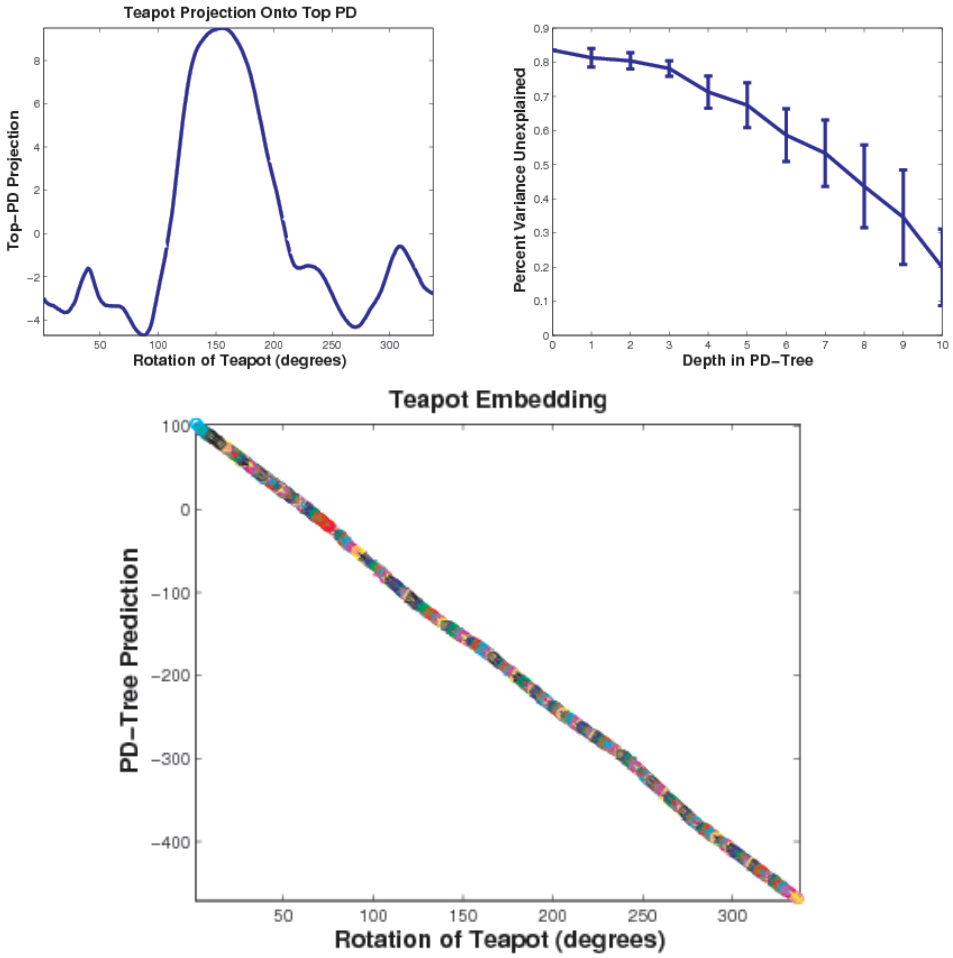


Figure 2.3: (Top-Left): Embedding of entire dataset on top principal direction. (Top-Right): Percent variance explained versus depth of PD-Tree. (Bottom): Embedding learned from a constructed PD-Tree.

Thus, the points x where $t_1 < x^\top v < t_2$ will go to both children nodes. This creates the needed overlap. We choose t_1 and t_2 so that the middle 30 percent go to both children.

We break the circularity of the dataset by dropping the last 25 examples, giving us 375 examples to train a PD-Tree on. This allows the embedding to be represented as a straight line in one dimension. We build a PD-Tree to depth 10 with the splitting procedure described in the paragraph above. As you examine nodes with increasing depth, the top principal direction becomes a better linear model of the associated teapot data. This phenomenon is depicted in Figure 2.3. It shows the average F value for nodes at a fixed depth in the PD-Tree with error bars indicating the standard deviation. It is

clear that as you descend the PD-Tree the sets become more and more linear.

We collected a set of good linear models by considering all internal nodes that have an F value below 0.3. This collection was then pruned to remove all nodes that have a large gap when $x^\top v$ are sorted. This is to remove those nodes that have points from disjoint regions that are physically far from each other. We used the ratio of the largest gap to that of the median gap value to be the pruning quantity, which we threshold at 4.

There are 781 total remaining sets where each datapoint $x \in S$ is covered by an average of 13.7 different sets. The embedding depicted in Figure 2.3 is formed by merging these sets in a greedy fashion. To start, we select the biggest set and embed its points with the projection value $x^\top v$. Then, we find the set that has largest overlap with the already embedded points (and is not strictly a subset of the already embedded points). We then orient the projections of these points so that the ordering of the projections match for both sets. This amounts to choosing a sign multiplier for the projections to flip their ordering. Finally, we solve for an offset to add to the projections we are going to merge into the embedding so that the overlap regions match closely. We repeat this process until all points are embedded.

The different sets are depicted in different colors in the Figure. To cover the entire dataset 208 sets were used, a large number, but not surprising since many of the original sets had a high level of overlap. The result is near linear. Moreover, this model can be extended to unseen datapoints points by averaging the models of the nodes in which they fall in. Thus, a simple scaling of the embedding value reveals the rotation of the teapot.

2.4 Case Example: TDOA Manifold

In this section we present another example of how a PD-Tree works on real data focusing on the case of the TDOA manifold. Recall that the time delay of arrivals (TDOA) between all pairs of microphones are the objects of interest in the audio localization models derived in Chapters 3 and 4. We consider an array arrangement of microphones that is near-planar, meaning all microphones are arranged on a single

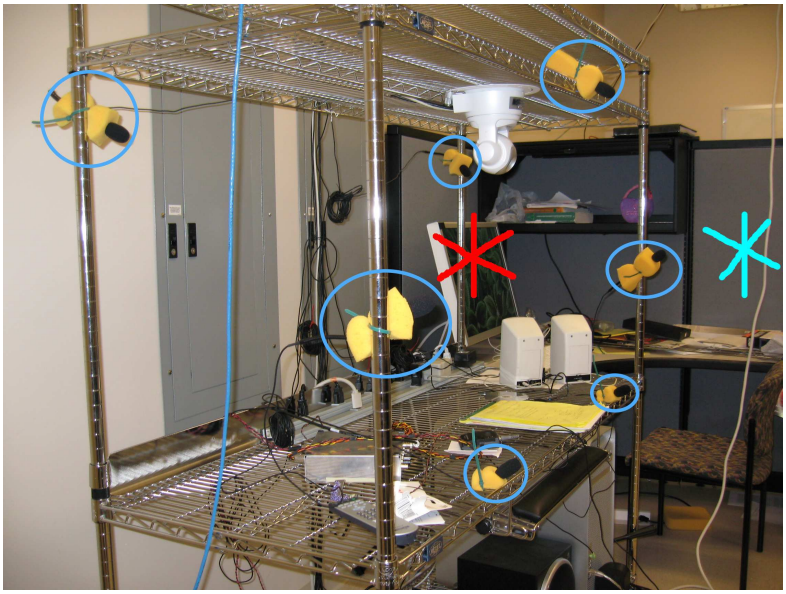


Figure 2.4: A photograph of the seven element planar array (microphones circled in blue). Two crosses are shown to illustrate the two cross-shaped datasets collected.

plane in space, in this case a vertical plane from floor to ceiling (they are only near-planar since they are positioned on a physical baker's rack with some at a depth of about 1 foot behind others). A picture of the array can be seen in Figure 2.4. This is one of the precursor's to the current version of TAC, when we first began researching the potential of passively locating sounds with a microphone array.

We collected a dataset by moving a small radio producing white noise throughout the region of interest in front of the microphone array. We then extracted the relevant TDOAs from this process by the method described in Chapter 3. This results in a dataset, S, of approximately 10k examples each of which is a 21-dimensional TDOA vector with coordinates representing the individual TDOA between a pair of microphones. The dataset was created by varying the physical 3-d location of the sound source. We first were interested in the ability of the array to differentiate between different locations in the room.

A microphone array can localize accurately in 3 dimensions when the sound source is inside the convex hull of the microphone elements. This is intuitive, since any variation in position results in a corresponding difference in the series of TDOAs observed across all pairs of microphones. However, for a planar array, judging depth from

the array is difficult since a change in depth does not always result in a significant change in the TDOAs observed. Thus, planar arrays are typically used to judge the azimuth and elevation of the sound source relative to the plane containing the microphones.

This intuition is confirmed by the experiment shown on the top-left in Figure 2.5. In this experiment, two small datasets were collected (after collecting S) by moving the radio producing white noise along each of the coordinate axes (the xy-plane oriented parallel to the floor). The motion traced by the radio is depicted by the two cross shapes in Figure 2.4. One of the datasets was collected inside the convex hull of the microphones, and one directly in front of the array. The resulting series of TDOAs collected are plotted onto the top 3 principal directions of S (capturing 99.8% of the variance in S). The dataset in front of the planar array is the one plotted in blue and shows no resolution in depth. The set collected inside the convex hull clearly has the ability to resolve depth.

We also grew a PD-Tree to depth 3 on S and depict the projection of the collected data set S onto its principal components in the top-right and bottom of Figure 2.5. The eight leaf nodes of the PD-Tree are depicted by color. Recall that S was collected in front of the array, so there is no depth resolution. Nevertheless, it's clear that the manifold is not linear by examining the plot on the top 3 principal directions. There is a clear curvature to the manifold, which we discuss further in Chapter 3. Thus, as we will see when we try to convert TDOAs into locations, a simple linear mapping to position from this projection space is not entirely desirable due to this nonlinear distortion.

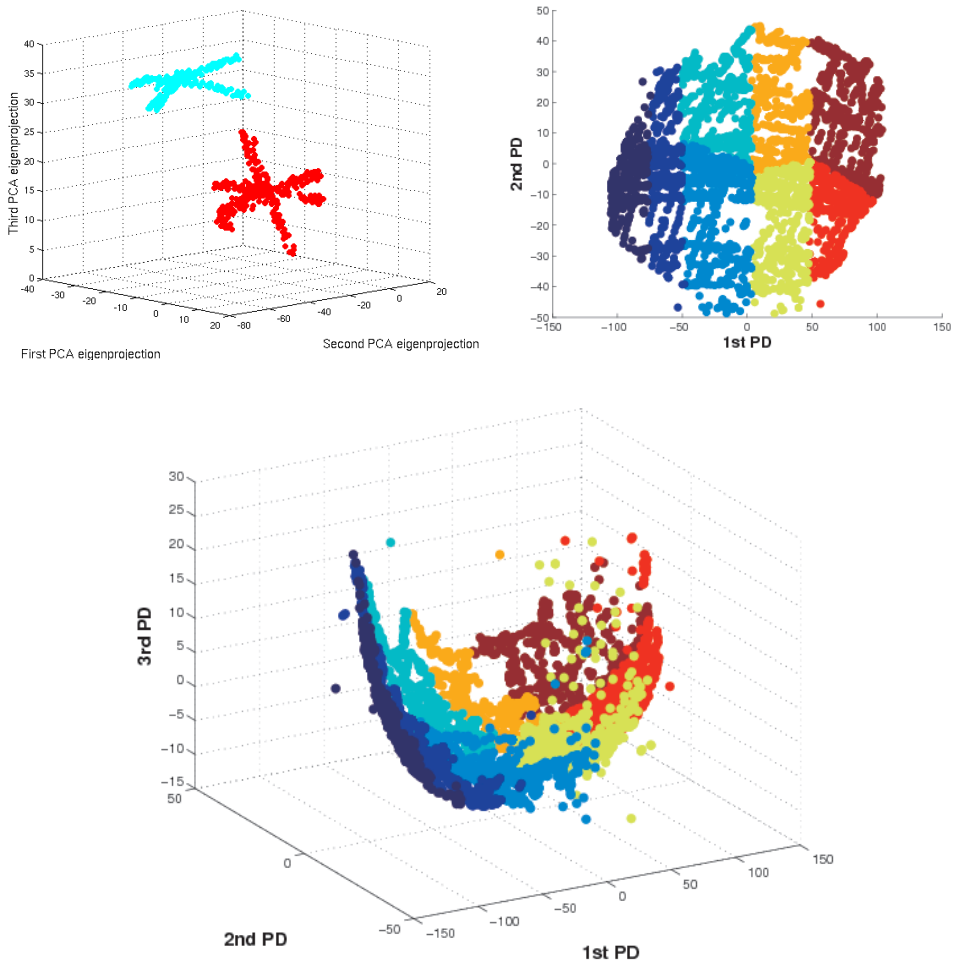


Figure 2.5: Top-Left: Cross datasets both inside (red) the convex hull of microphones and outside (cyan). Top-Right: Embedding of entire dataset on top 2 principal directions; leaf node membership is color coded. Bottom: Embedding on top 3 principal directions.

Chapter 3

Audio Localization

In this chapter we explicitly outline the methodology used to first estimate time-delays and then map them to pan-tilt directives for camera pointing. The methodology differs from previous research in that the localization is done in a *coordinate-free* fashion. That is, we do not explicitly need to know the positions of the microphones or the position/orientation of the PTZ-camera.

The organization of this chapter is as follows. First, we discuss the standard methodology and theory of time-delay estimation, namely, the phase-transform correlation method. Second, we discuss various existing techniques for doing sound source localization. As we will see, these methodologies do not explicitly address the problem of camera pointing and they all rely on the a priori knowledge of the coordinates of microphone elements to solve the localization problem geometrically.

As an alternative approach, we propose a machine learning solution to the sound localization problem based on the collection of a training set from which a regressor can be learned. The regressor will map time-delays to pan-tilt directives for pointing the camera, skipping the need for a coordinate system in which microphone elements are known. We outline this coordinate-free approach and show experiments with a dataset collected from TAC showing less than 4 degree error in both pan and tilt – results competitive with the current state of the art. The majority of work found in this chapter can be found in the published version here [EF08].

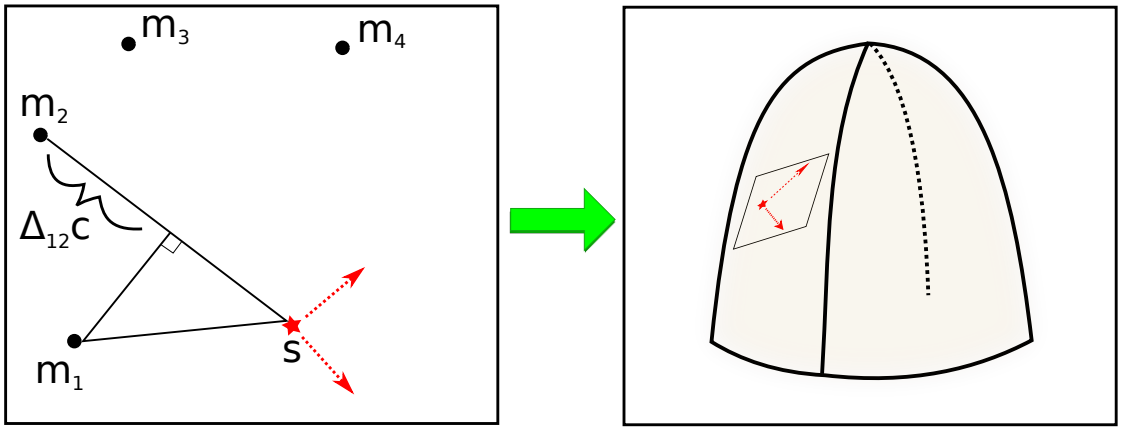


Figure 3.1: Left: A 2-dimensional world with 4 microphones. Time-delay Δ_{12} is shown between microphones m_1 and m_2. The sound source (red star) is shown with 2 degrees of freedom for movement (red arrows). Right: Depiction of the 2-dimensional manifold created by the sound source's movement. The corresponding local movement from the figure on the left is shown as a locally linear region of the manifold.

3.1 Time-delay Estimation

Recall that the fundamental basis that allows a microphone array to localize a sound source is the time-delay of arrival (TDOA) between two spatially separated microphones. Leveraging this physical fact is the fundamental insight behind passive sonar technologies. An illustration of the TDOA produced from a 2-d world is shown in Figure 3.1.

Even though in this work we do not assume knowledge of microphone or camera positions, it is useful to assume they are known and fixed for the discussion that follows. Let $m_i \in \mathbf{R}^3$ be the three dimensional Cartesian coordinates for microphone i. For a sound source located at position s and assuming a spherical propagation model, the direct path time delay between microphone i and j can be calculated as

$$\Delta_{ij} = \frac{\|m_i - s\|_2 - \|m_j - s\|_2}{c} \tag{3.1}$$

where c is the speed of sound in the medium. Δ_{ij} is often called the *time delay of arrival* (TDOA) between microphone i and j. It is worth noting that if f is the sampling rate being used, then the largest the TDOA can be in terms of audio samples is $M = \|m_i - m_j\|_2 f/c$. In other words, Δ_{ij} is always in the range $[-M, M]$ and in practice can

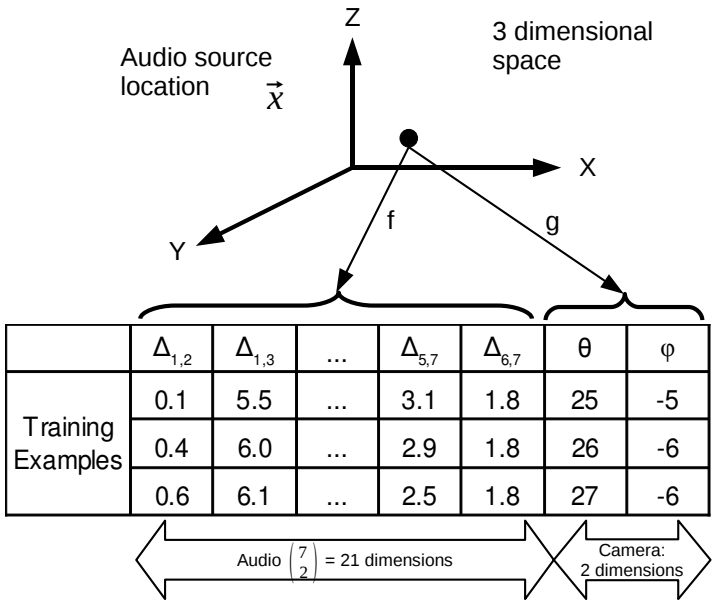


Figure 3.2: The sound manifold. f is a mapping from sound source location x to a set of TDOA measurements $\vec{\Delta}$. g is a mapping from x to a pan and tilt directive for the PTZ camera.

only be estimated to the nearest sample. This observation directly reveals the fact that close together microphones cannot have as wide a range of TDOAs as microphones that are spaced further apart. Placing microphones further apart allows for more variability in the feasible TDOAs, and hence, results in a better ability to discriminate between audio source locations in space.

Given k microphones there are $\binom{k}{2}$ unique pairs of microphones for which Δ_{ij} can be estimated. We let $\vec{\Delta} = (\Delta_{ij})_{i<j} \in \mathbf{R}^{\binom{k}{2}}$ be the vector that contains each of these unique TDOAs for a given audio source location. We will often call $\vec{\Delta}$ the *TDOA vector* (see Figure 3.2). For a $\vec{\Delta}$ that corresponds to a true audio location there are many linear dependencies between the components because

$$\Delta_{ij} = \Delta_{kj} - \Delta_{ki} \quad \forall i, j, k \tag{3.2}$$

Therefore there are only $k-1$ linearly independent coordinates of each $\vec{\Delta}$. In other words, the pairwise delays to just a single reference microphone uniquely determine the

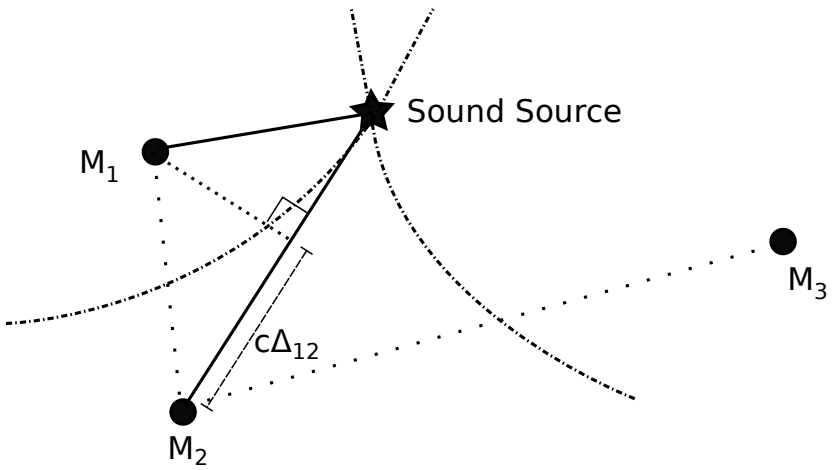


Figure 3.3: A 2-d world where 3 microphones are necessary to uniquely determine a sound source's location via multilateration. If given Δ_{12}, Δ_{23} and knowledge of the microphone positions, then one can solve for the intersection of the corresponding hyperbolas for s.

delays for all pairs. Nevertheless, when estimating $\vec{\Delta}$ it is useful to consider all $\binom{k}{2}$ pairs since the estimation procedure that we will employ is fast to compute, but often noisy.

When given a fixed Δ_{ij} for a pair of microphones, we can deduce from Equation 3.1 that the set of feasible s positions that could have resulted in the observed Δ_{ij} form one sheet of a 3-d hyperboloid in space (for a 2-d world representation see Figure 3.3). It follows that for a fixed $\vec{\Delta}$, the possible audio source locations that could have generated such a TDOA vector can be determined through finding the intersection among all such hyperboloids. This procedure is historically known as *multilateration*, which is similar in spirit to trilateration: instead of TDOAs, trilateration measures absolute time-of-flight to several beacons allowing for an intersection of spheres to occur. In multilateration, it is only required that we observe three TDOAs (4 microphones needed) to uniquely determine where in space the sound source is under an idealized estimation procedure.

However, in practice we can only estimate each Δ_{ij} from the underlying audio signals. As a result, the estimation procedure faces multiple challenges that easily lead to inaccuracies. First and foremost, sound easily bounces off of many physical materials causing multipath reflections and reverberations. Reflective materials such as linoleum

floors and windows are present throughout the room in which TAC is built. Secondly, the audio signal is only captured at a finite precision with respect to time since the signal must be digitized with a finite sampling rate. This means we can only estimate TDOAs with a finite precision the depends on the audio sampling rate.

These challenges often results in estimation errors in Δ_{ij} and so it is not surprising that in practice the intersection of all the corresponding hyperboloids is empty! In response, most methods employ an optimization framework like that of least-squares to find a source location that is close to all hyperboloids. This has been the focus of a lot of research and a discussion can be found in [JD01]. Including more microphones in the array adds a redundancy to the information content of $\vec{\Delta}$, which can be leveraged to make a more robust localization system.

Accurate and robust time-delay estimation (TDE) is the key to many types of localization systems. Background noise, multipath reflections and room reverberations complicate the estimation process. There has been much research on TDE in a variety of fields, and a review of many techniques can be found in [JD01]. Nevertheless, one very intuitive way to estimate a TDOA would be to calculate the cross-correlation of a pair of microphone signals and find the maximum. Unfortunately, due to corrupting factors of the signal this often gives maxima that are not near the true TDOA.

One of the most popular TDE techniques, and the method used in this work, is a generalized cross-correlation (GCC) technique that utilizes the phase transform (PHAT), first discussed in the audio localization literature by Knapp and Carter and then further analyzed by many others [KC76, OS94, OS96]. PHAT is very robust to noise and reverberations compared to other correlation based TDE techniques [JD01, SMO97]. Let $X_k(\omega)$ be the Fourier transform of microphone k. The GCC between microphone l and m is

$$R_{lm}(\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \Psi(\omega) X_l(\omega) X_m^*(\omega) e^{j\omega\tau} d\omega \tag{3.3}$$

where $\Psi(\omega)$ is a weighting function for the GCC and * denotes complex conjugation. The PHAT weighting of the GCC is of the form

$$\Psi(\omega) = \frac{1}{|X_l(\omega) X_m^*(\omega)|} \tag{3.4}$$

The PHAT weighting has a whitening effect by removing amplitude information in the

signals. Compared to standard cross-correlation, PHAT puts all the emphasis on aligning the phase component of the transformed audio signals and none on the amplitudes. Empirically, it has been observed that the result of using the PHAT weighting is often a large spike in the GCC at the true TDOA. Hence the PHAT method for TDOA estimation is to let

$$\Delta_{ij} = \arg\max_{s} R_{ij}(s) \tag{3.5}$$

In Figures 3.4 and 3.5 we show results from both traditional cross-correlation (unweighted) and PHAT correlation for an audio segment recorded with TAC. The PHAT correlations have been verified to have a maximum corresponding to the true TDOA of the human speaker. As can be seen, the unnormalized cross-correlation is smooth, but often can have a peak that is incorrect. In fact, only one component (Δ_{16}) would be estimated correctly using standard cross-correlation for this frame of audio. The PHAT correlations are typically very pronounced at the estimated TDOA with a small number of significant secondary peaks.

In TAC, we utilize a sampling rate of 16 kHz, a window of 500ms and a step size of 25ms for calculating the PHAT correlation. C code for calculating the PHAT correlations through an FFT was generating using Simulink in MATLAB [Sim].

One of the primary competing forces in time-delay estimation is between placing microphone elements near to each other or far away. Placing elements near each other gives incoming signals that are very similar and hence result in more accurate TDE. However, placing elements farther apart allows for both increased spatial discrimination and coverage of a room. The system designer must weigh these competing forces in conjunction with the number of microphone elements available when constructing such an audio localization system.

In this work we are interested in an ad-hoc placement of both the microphones and the camera. We do not assume knowledge of any m_i or the location and orientation of the PTZ-camera. Instead we would like to learn from a training set how to direct the camera for a given $\vec{\Delta}$. We would like to learn a regression function that describes how the pan and tilt of the camera changes with variations in $\vec{\Delta}$. Before we dive into the development of such a regressor, it would be worthwhile to understand some features of this variation so that we can reasonably select a regression model that can capture such

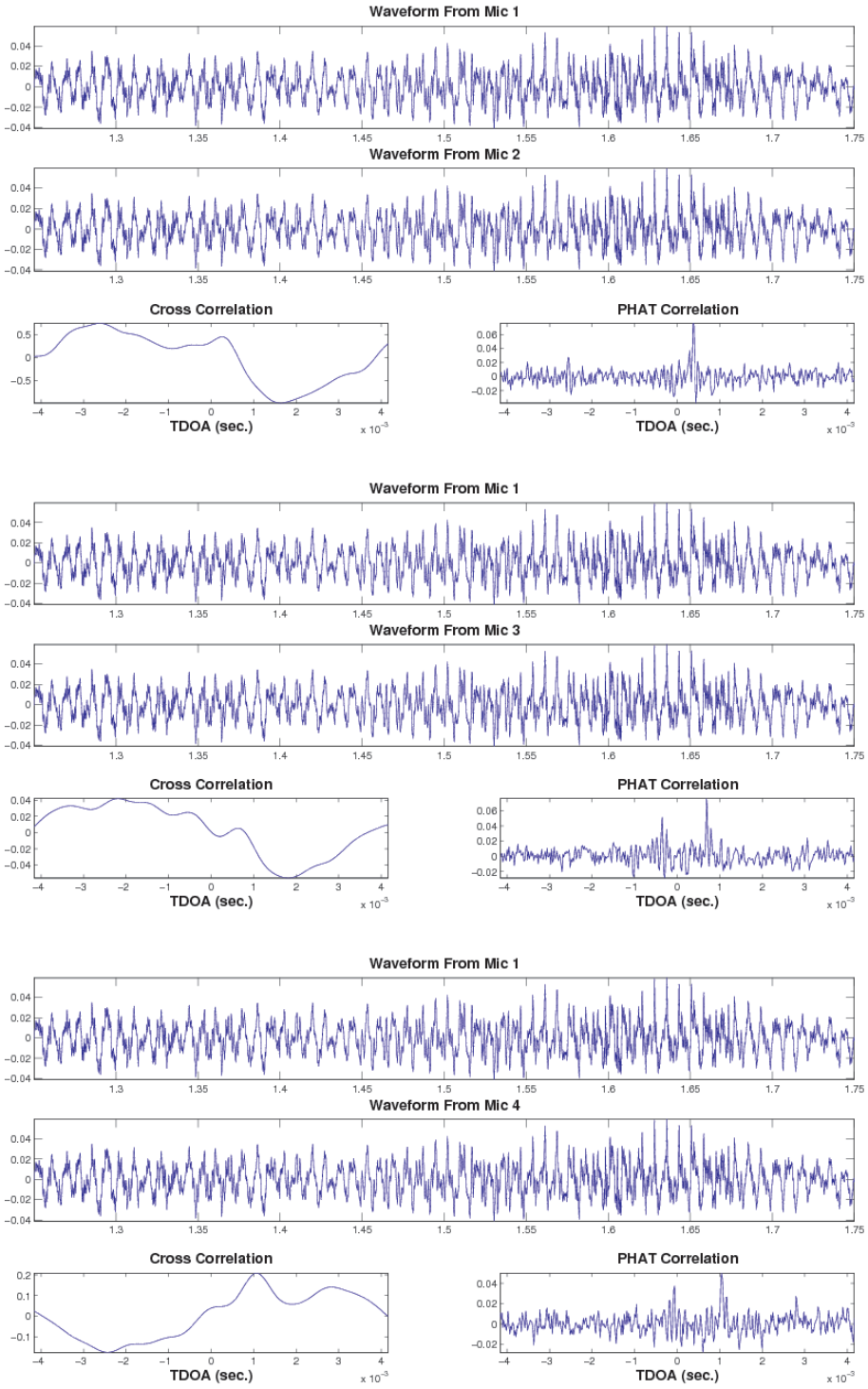


Figure 3.4: 500ms real audio sample collected from TAC with corresponding cross-correlation and PHAT correlation results. Pair combinations with microphone 1 are considered: Δ_{1j} for $j = 2, 3, 4$.

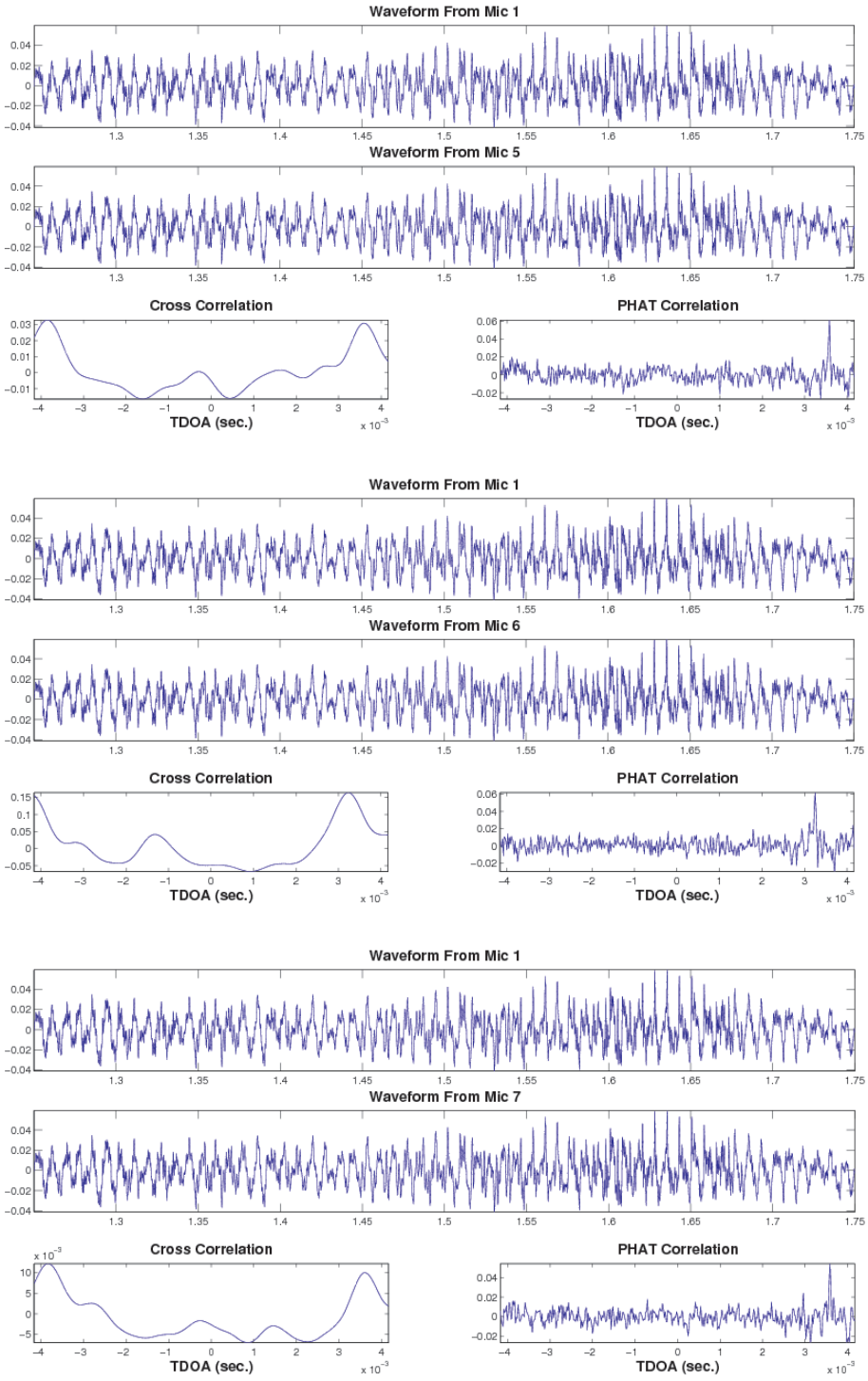


Figure 3.5: Continuation of Figure 3.4. Pair combinations with microphone 1 are considered: Δ_{1j} for $j = 5, 6, 7$.

behavior.

Notice that when a speaker is close to the camera, small deviations in position correspond to large deviations in pan and tilt when compared to the same sized movements when far from the camera. Therefore, it would be reasonable to believe that the predictive function from TDOAs to pan and tilt is most nonlinear for TDOAs that correspond to locations close to the camera. Moreover, notice that when nearby a pair of microphones small movements in position correspond to large changes in the TDOA relative to the changes in TDOA when far way. From this discussion, it's likely to believe that a linear model will not capture all the variation between these two sets of variates, but in areas where the variations in each match, a linear model may be very accurate. Moreover, it is not clear how poor a linear model would be overall and it is unclear at a surface level how much nonlinearity to expect. This motivates the inquiry into very simple regression models to fit this variation and then the examination of how and where these models perform below expectations. In this chapter we will examine the performance of global regressors such as linear regression and also how piecewise models constructed via PD-Tree models perform as well.

3.2 Related Work

Sound localization techniques via microphone arrays can be summarized into two major paradigms: TDOA two step localization and steered response power (SRP) based. The first technique involves first estimating for a frame of audio the TDOAs between all pairs of microphones and then solving the subsequent geometric multilateration problem. Many of the approaches utilize similar TDOA estimation techniques as the PHAT approach discussed in the previous section. The difference lies in the approach to the geometric solution. The most popular is a least squares approach. One such approach is to simplify the nonlinear least squares problem by linearizing it through either a Taylor expansion [Foy76] or by introducing an extra variable as a function of the source location [Fri87, SA87, CH94, HBEM01, SL06, GS08]. This leads to a closed-form solution to the problem since it becomes a linear least-squares problem, but the resulting variance in the source location estimator is large [CH94, HBEM01].

Other approaches attack the nonlinear least squares problem directly. Silverman et. al use the simplex algorithm to find the the best match for the observed TDOA from the theoretical TDOA caused by a sound source at a fixed 3-d location [SYSP05]. Brandstein et. al use the popular line-intersection method that employs the far field assumption to find a point closest to all the bearing lines originating from all pairs of microphones [BAS95]. Gustafsson and Gunnarsson compare many different schemes: a simple weighting of the pairwise intersection points, a stochastic gradient descent based algorithm, and one based on a particle filter [GG03]. Many other techniques exist in this two-step category as well.

The second category for source localization techniques are all based on maximizing the steered response power (SRP) of a beamformer [JD01]. For example, a simple instance in this class is to maximize the energy of a delay-and-sum beamformer over a range of steering directions. That is, for each source location x, one can first calculate the corresponding TDOA vector, $\Delta(x)$, derived from the array geometry. By delaying the frames of audio by these TDOAs and then summing all the signals together, the energy of the signal can be estimated. The underlying assumption with an SRP based localization method is that the energy will be maximized at the best choice for location x. Probably the most popular of SRP based beamformers is the so called SRP-PHAT beamformer [JD01, DSY07]. Here, instead of maximizing the energy of the delayed signals, one calculates the PHAT correlation, $R_{ij}(\tau)$, for all pairs of microphones and then solves the optimization $\arg\max_x \sum_{i<j} R_{ij}(\Delta_{ij}(x))$.

One advantage of SRP-PHAT is that it is more robust against the failure points of PHAT estimation, namely false peaks due to multipath reflections. Often when the true TDOA is not the largest peak it is still among the few number of large peaks present in the PHAT correlation. As a result, it's contribution to the SRP-PHAT objective is still large. Moreover, it is often the case that the largest peaks from each pair cannot even produce a feasible TDOA vector that correspond to an actual source location. Thus, a naive implementation would simply do a search over potential source positions $x \in \mathbb{R}^3$, typically by gridding the area of interest. However, for applications that require both real-time and precise positions a simple grid approach is often too slow. Therefore, optimizations like "coarse-to-fine" are popular where a large grid is exhaustively searched

and then neighborhoods are refined at locations with a large SRP-PHAT value [JD01].

Both the two step and beamforming based methods require knowledge of a coordinate system wherein microphone positions are known. For small microphone arrays a coordinate system can easily be found by simply measuring the distances between microphones by hand as in [WC97]. If we want to be able to localize sounds in a large room accurately, then a large microphone array that spreads throughout the room is beneficial. However, measuring accurately by hand the relative distances now becomes much more difficult and positional errors on the order of 1-5cm can seriously degrade beamforming techniques [SSP05].

Since doing such measurements is often too difficult, especially for arrays with many elements, many techniques have been developed to automatically calibrate the positions of the microphone elements [SSP05, RD04, BS05, MLH08, HLKB05]. These techniques are based on using a carefully designed device that emits a special sound with which delay measurements and the known geometry of the device can be leveraged to solve for the microphone positions. Typically distances from the device to the microphones, or inter-microphone distances are estimated. For example, if pairwise distances between microphones can be estimated, then traditional multidimensional scaling (MDS) is often used [SSP05, RD04, BS05, MLH08, HLKB05].

These techniques are intended for a one-time calibration of a system with positionally static elements. Shifts in microphone positions can occur for a variety of reasons especially if they are placed in an ad-hoc manner. These positioning methods do not have self-consistency checks for continued positional accuracy during general usage. More importantly, such calibration techniques are not geared towards camera steering. They gives us no insight into methods that let us place and orient a camera in the same coordinate system.

In this work, we avoid the need for solving for microphone (or camera) positions explicitly and are still able to utilize the benefits of much of the sound localization research community's work. Our intent is to develop a robust system for pointing a pan-tilt-zoom (PTZ) camera at sound sources in front of an interactive kiosk in a large room. If we were to directly use current sound localization techniques, then we would be required to discover the coordinates of not only the microphones, but also of the PTZ

camera. This would require either direct measurement or new calibration methods to locate and orient the camera.

Instead, we curtail the need of assigning the camera and microphones spatial coordinates by directly learning the mapping from the set of delays for pairs of microphones to the correct pan-tilt (PT) of the camera so that the sound source is centered in the field of view. We do this by collecting observations consisting of a set of delays between microphones for a fixed source location and the associated PT to center such a source. With this database of samples, we estimate via standard regression analysis a fixed model for the system. This model describes how PT and delays vary with each other. The result is a function that can map a series of delays between pairs of microphones to a PT directive for our camera. It is very natural to then combine this mapping with known audio localization techniques. Together this gives us a real-time implementation that can direct the camera at human speaking subjects.

3.3 Coordinate-Free Localization

In this section we describe the regression models we decided to use for describing the variation between $\vec{\Delta}$ and pan-tilt directives. For what follows assume that a training set of size m is given with observations of the form $y_i = (\theta_i, \psi_i)$, for pan and tilt respectively. These observations are paired with an estimated TDOA vector derived from the N microphones, namely $x_i = \vec{\Delta}_i$ with $p = \binom{N}{2}$ coordinates. We organize the training set into matrices $Y \in \mathbf{R}^{N\times 2}$ and $X \in \mathbf{R}^{N\times p}$ where each observation is a row vector. In what follows, we briefly remind the reader of least squares linear regression and a variation called principal components regression. Further information on both can be found in [HTF01].

Least Squares Linear Regression

For each column of Y, denoted Y_i, we fit a separate linear regression model. The linear regression model has the form

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

where X_j is the j^{th} column of X and β is the vector containing the coefficients in the linear model. The least squares (LS) solution to linear regression chooses the model that minimizes the residual sum of squares (RSS)

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

When X is full rank the LS solution can be written in closed form as $\beta = (X^T X)^{-1} X^T Y_i$. It is known that if the true model of data generation is linear, then the LS estimator is the minimum variance unbiased estimator of β.

Principal Components Regression

Often in regression it's advantageous to trade a small amount of bias for a large reduction in variance. Principal components regression (PCR) attempts to describe the $k < p$ orthogonal directions in the feature space that preserve most of the variance in X. After centering X, PCR projects the data onto the k-dimensional subspace spanned by these directions, and learns a LS linear regression model to predict Y in this reduced space. Although the resulting model is slightly biased if the true underlying model is linear in the original feature space, typically the reduction in parameters by PCR results in a dramatically smaller variance in estimating its parameters and hence a corresponding smaller RSS. In addition, an attractive feature of PCR is it's denoising properties of X. By projecting it onto the learned subspace, noisy coordinates in the TDOA vector may be improved through this denoising procedure.

Calculating the top principal components is achieved through principal components analysis (PCA). The first principal component v_1 is defined as the direction in the feature space that gives the projections of X onto it the highest sample variance. Subsequent principal components have the property that when X is projected onto them they have the next largest sample variance subject to being orthogonal to all previous principal components. Solving for the principal components can be shown to be solved by an eigendecomposition of the covariance matrix $X^T X = VDV^T$. The column of V with corresponding largest eigenvalue is the first principal component, and the eigenvector with next largest eigenvalue is the second, and so on. The percentage of variance in X explained by a principal component is the ratio between its corresponding eigenvalue

and the trace of D. Typically one retains the top k principal components that describe most of the variance and discards the remaining deeming them as observation noise.

Higher Order Polynomial Fits

We can fit general polynomials using the LS approach by simply extending X to contain higher-order combinations of features. For example, in the quadratic regression (QR) analysis used in the experiments that follow, appending to X the squares and cross-terms of features and applying the LS method gives the desired parabolic fit. The same procedure can be repeated for the data matrix X used in PCR.

PD-Tree

In the experiments that follow we will also explore the use of a constant depth PD-Tree (see Chapter 2) with regressors learned in each leaf node. This will act as a piece-wise regression model. We grow a PD-Tree to depth 2 and fit linear least squares regressors in each leaf node.

3.4 Experiment: Grid Dataset

The device used to collect all the data in the experiments to come is shown in Figure 3.6b. It consists of a simple radio and a green LED attached to a 9V battery with a switch and dimmer all in a plastic encasing. We will call this the *calibration device* from here on. The radio component of the calibration device can be tuned to a nonexistent station that emits noise that is very close to white. This random noise typically has the most consistent TDOA vector estimates using the PHAT technique. A simple color thresholding detector was written to find the LED in the camera's field of view using Max/MSP and Jitter [max]. The result is a real-time control of the PTZ-camera to keep the LED centered in the field of view, and a constant white noise to calculate TDOAs for. The calibration device is used to collect samples of TDOA vectors in unison with where the camera is pointing to center the green LED in its field of view. The camera can be queried as to what pan and tilt it is currently pointed at whenever a TDOA vector

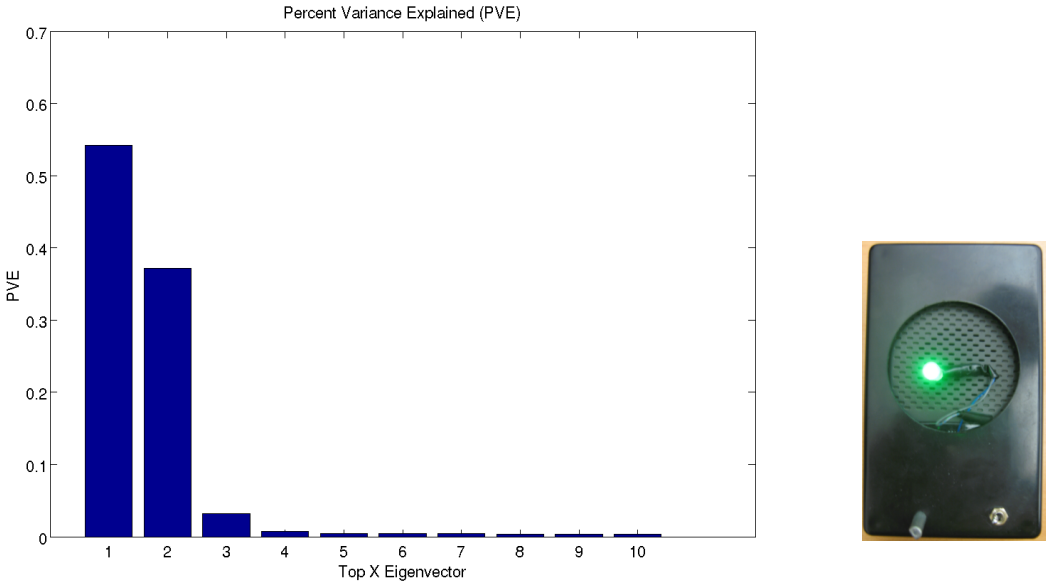


Figure 3.6: Left: Percentage of variance explained by top X eigenvector. The top 3 eigenvectors dominate and the rest are noise. Right: Calibration device used to collect training and grid dataset.

is collected to append this information as a complete data observation.

The result of the training set collection is a dataset of close to 28k observations. We noticed that when a estimate for Δ_{ij} was incorrect, it typically had a very large deviation from what was often consistent. To remove such noisy observations, we performed some simple outlier removal by thresholding the magnitudes of the $\vec{\Delta}$ projections onto the bottom global PCA eigenvectors (orthogonal space) leave approximately 20k observations remaining as our training set. We then did a PCA analysis of just the $\vec{\Delta}$ parts of this training set. Figure 3.6 shows the percentage of variance explained by the addition of each eigenvector. It's clear that the top two eigenvectors dominate most of the variance explained, and that the 3rd eigenvector seems to have a significant advantage over the remaining ones. The total percent variation captured by the top 3 eigenvectors is nearly 90%. This follows from the fact that there are 3 spatial degrees of freedom that were examined during the training data collection period. Moreover, two of these spatial directions had much more spatial variance then the third, ceiling-to-floor, spa-

tial direction. The room is simply much larger in width and breadth than the variance in observation heights, which matched typical heights that human speakers could appear at. This analysis lead us to choose three eigenvectors for the principal components regression analysis that follows.

From this training set with outliers removed we have nearly 20k observations. From here we learn a simple linear least-squares regression (LS) model, a linear principal components regression (PCR), a quadratic least-squares regression (QLS) and a quadratic principal components regression (QPCR). We would like to analyze how the bias-variance trade-off of these simple models behaves as function of physical position of the sound source in the lobby. In other words, in what areas do these simple models perform well, and where does the inherent nonlinearity of the problem cause large bias?

With these questions in mind we collect a test set of data in a similar fashion to the training set. We place the calibration device at a fixed height (approximately 1m from the floor) and roll it along straight lines using a rolling chair. We repeat this process for each of the 13 lines in the grid depicted in Figure 3.7b. This results in a variety of observations that cover a representative set of the spatial variability in the room relevant for human speakers. Moreover, using white noise as our sound source will simulate the behavior of our model under conditions where TDE is highly optimized. This gives us insight into isolating the effects of the model assumptions.

Each of the 13 grid lines was traversed back and forth two times during the collection phase. This test set of observations collected along the grid will be used to analyze the predictive power of each of the regression models. It is worth noting that although using the light detector will not give an exact "ground truth" comparison, it is nevertheless very close and thus a fair evaluation. The light detector observations for pan and tilt are very consistent and stable when the calibration device is stationary. The camera was directed to recenter the LED whenever the center of color thresholded pixels exited a small 20 by 20 pixel box in the center of the image. Therefore, the pan and tilt observations from this test set should be considered as very close to having the sound source centered, which is the ultimate goal of this camera pointing system.

Figure 3.7a depicts the embedding of the TDOA vector components of the entire grid test set onto the top 2 eigenvectors from the PCA learned from the training set. The

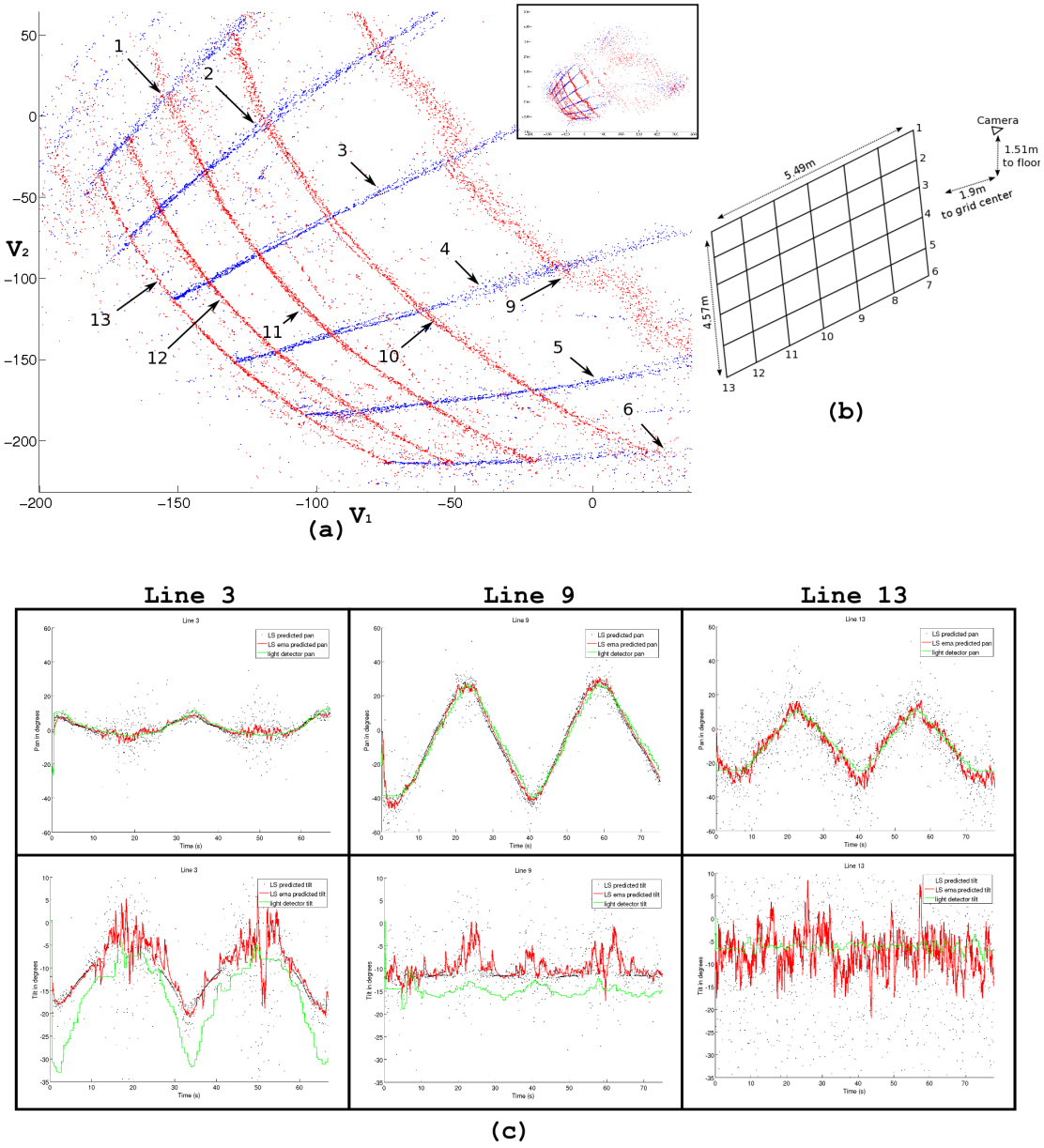


Figure 3.7: (a) Embedding of the TDOAs collected from the grid onto top 2 eigenvectors. The entire embedding is shown small in the upper right corner and a zoomed in portion of the same embedding is shown larger. (b) To the right is a diagram of the equispaced grid over which data was collected. (c) Below are 3 selected lines and the LS predicted value for each TDOA collected. Also depicted in red is an exponential moving average of the predictions ($\alpha = 0.10$), and in green where the camera was pointing to center the LED.

zoomed in portion depicts lines 9-13 in red and lines 1-6 in blue in the same orientation as the diagram in Figure 3.7b. The curved nature of each line can be observed from such plots. Even though the spatial location of the sound source is varying along a straight line in space, the corresponding location in the TDOA vector space corresponds to slightly curved trajectories. It is clear that a linear model for spatial location is not going to fully capture all the variation, but nevertheless the grid structure is still very recognizable in even just the top 2 eigenvectors indicating that a linear model is a good approximation in these regions.

Another thing to observe, especially in the smaller full plot of the entire embedding, is the noisy nature of the observations themselves. Although, the majority of the TDOA vectors are estimated along trajectories, there is quite a bit of noise. This is attributed to the noisy nature of time delay estimation; although the majority of TDOAs consistently follow a fixed trajectory, observations are occasionally noisy due to channel corruptions or reflections from room surfaces. This noisiness highly depends on location. For example, the observation noise from lines 1 - 6 increases as a function of distance from the display, and hence the microphones. It is also interesting to see that lines 7 and 8 are particularly noisy. This is most likely due to the fact that these lines are still in front of the microphones on the ceiling, and the data was collected while the radio faced the display. This causes the direct path to these ceiling microphones to be not as strong as reflections of the floor or front wall. This resulted in a variety of TDOA estimates for pairs involving the ceiling microphones at these particular locations.

Figure 3.7c compares the predictions from the simple linear LS model to the pan and tilt recorded from the light detector. The dots in black are the predicted pan (or tilt) from the model for each TDOA vector observation. The green line depicts the pan (or tilt) from the light detector. Finally the red line depicts an exponential moving average (EMA) of the model predictions over time. In other words, the EMA prediction, p_t, at time t is calculated with update $p_t = (1-\alpha)p_{t-1} + \alpha f(\Delta_t)$, where $f(\Delta_t)$ is the prediction of the raw observation at time t. We chose $\alpha = 0.1$. The EMA line should give us a sense of what the true model predictions are by smoothing out the observation noise. In doing so, we can compare the light detector observations to the EMA line and get a sense for the bias in our model.

Remember that for each grid line we collect data along two round trips across the line, which is why we see the periodic nature in the data. It's also worth noting that the light detector observations are slightly lagged from the truth. This is because the camera only recenters after the light detections exit a 20×20 pixel box in the center of the image. It's easy to see the bias of the model due to the nonlinear nature of some of the variation in these plots. For example, in the line 3 plot for tilt, the slop of the EMA line does not match the rate of change from the light detector. The attempt to capture the portions of tilts that occur when a sound source is close to the display is modeled to closely to constant when this is clearly not the case. However, this bias shrinks for lines that are further from the display.

It is also worth noting that as you move further from the display the TDOA vectors themselves become more noisy, which can be observed in the plots for line 13. Nonetheless, the bias in tilt is still the most dramatic for most grid lines, since this straight line motion in in space does not correspond to linear changes in tilt. On the other hand, for pan the changes are very near to linear.

Table 3.1 gives the root-mean squared error (RMSE) between the EMA of the model predictions and the observations from the light detector for each of the regression models. The overall averages are very similar to results reported by traditional coordinate based methods, meaning that coordinate-free methods need not sacrifice accuracy [BVMA09].

Surprisingly, the PCR methods do not show any advantage over their LS counterparts, and in fact are significantly worse. There is no advantage for trying to remove additional noise from the observations. The variance in estimating the LS model is most likely very low because of the large quantity of observations collected in the training set. Moreover, the bias in the both the LS and PCR models should be similar because they are both linear approximations, giving the LS model an advantage in total RMSE. The poorer performance of PCR can only be attributed to the fact that some signal is being removed by projecting down to only the top 3 eigenvectors and not the entire space.

We also provide information about quadratic models of both. The quadratic models do show a slight improvement over the linear model in almost every line, which indicates that significant improvement with nonlinear models is possible. A similar

Table 3.1: RMSE (in degrees) of different regression models for each grid line.

Model	**Grid Line Number**													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**avg**
LS-pan	4.31	3.34	2.77	2.26	2.22	4.33	5.99	6.54	3.56	3.95	3.20	3.83	3.96	3.87
PCR-pan	7.30	5.08	4.67	5.07	4.78	5.60	6.85	6.26	4.54	4.86	4.99	8.24	8.04	5.87
QLS-pan	4.25	3.12	2.50	2.08	2.10	3.47	4.32	5.46	3.12	3.50	2.09	3.03	3.35	3.26
QPCR-pan	7.55	4.49	3.95	4.41	4.27	4.81	5.47	5.76	3.53	4.47	4.30	8.07	7.42	5.27
PD-pan	4.22	3.45	3.14	1.87	3.05	3.73	4.14	5.08	3.05	3.45	2.45	3.61	3.88	3.47
LS-tilt	5.15	5.74	7.57	7.67	7.50	5.48	3.33	8.40	5.63	4.74	3.90	5.15	4.48	5.75
PCR-tilt	4.02	6.83	9.13	9.23	9.27	6.17	2.65	11.12	6.61	5.21	3.03	2.29	3.37	6.07
QLS-tilt	4.43	4.40	4.32	4.32	4.47	4.23	3.06	3.14	3.32	3.23	2.17	4.59	6.13	3.99
QPCR-tilt	5.21	5.62	6.47	6.25	6.53	5.95	3.83	7.66	5.72	4.61	2.43	3.75	5.72	5.36
PD-tilt	4.70	4.85	4.72	4.68	4.65	4.76	3.26	3.51	4.82	4.42	2.95	5.25	6.55	4.55

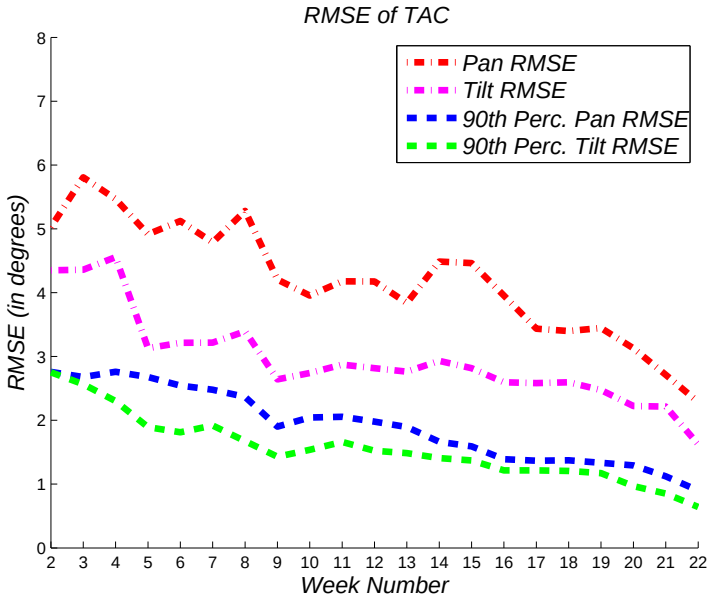


Figure 3.8: RMSE for pan and tilt of a PDTree trained each week with new data acquired by TAC.

phenomenon can be seen in the PD-Tree results. This piecewise model also outperforms a simple global linear model.

Because of inaccuracies from the lag of the light detector the RSME results here should be treated as an upper bound of the L1 error in pan and tilt. With a QLS model we can get almost uniformly throughout the room within four degrees error in both pan and tilt. This kind of uniform bound implies that predictions made for distances that are farther form the display have higher spatial error than of predictions for sources that are closer. It is a nice result that such a simple model of the variation can work well enough for many practical applications.

3.5 Experiment: Lifelong Learning

We can easily acquire a training set to learn a regressor with a little help from the face detector. Training examples can be collected whenever a user speaks while their face is centered in the field of view, creating a stable measurement of the form $(\vec{\Delta}, \theta, \phi)$. Many such examples can be collected over time by having the PTZ-camera continually

centering the user's face and the user continuing to speak. This is in fact what we do in TAC. Whenever a user is interacting with TAC a log is recorded that records these stable training points. We retrain a PDTree with linear models in the leaves at the end of each week on the entire training set collected up to that point.

We took all the observations TAC has seen over a period of approximately 6 months ($\sim$3000 observations), and split this randomly into a 70/30 training and test set. We then examined how TAC can improve its localization accuracy by retraining a regressor for pan and tilt each week on the data from the training set seen to that point. We averaged root-mean squared error (RMSE) calculations over 20 such random training/test splits. Figure 3.8 shows the improvement of this regressor in terms of RMSE. Also shown is the RMSE when the top 10% of squared-residuals are removed from the RMSE calculation.

The improvement is near-linear from week to week. Moreover, many of the errors are near or below one degree in both pan and tilt. This is promising since the locations in the test set are representative of where most users frequent when interacting with TAC. This means we are very accurate ($<$ 1 degree error) in these locations.

Acknowledgements

Chapter 3 is based on two works. The first is joint work with Yoav Freund entitled "Coordinate-Free Calibration of an Acoustically Driven Camera Pointing System" appearing in the proceedings of the International Conference on Distributed Smart Cameras in 2008. The dissertation author was the primary investigator and author of this work. The second is joint work with Sunsern Cheamanunkul, Matt Jacobsen, Patrick Lai and Yoav Freund titled "Detecting, Tracking and Interacting with People in a Public Space" appearing in the 11th International Conference on Multimodal Interfaces and 6th Workshop on Machine Learning for Multimodal Interaction in 2009. The dissertation author along with Sunsern Cheamanunkul were the primary investigators and authors of this paper.

Chapter 4

Tracking

In the previous chapter we discussed methods in which TDOAs can be estimated and used to predict directives for camera pointing towards a sound source. The methods discussed there are coordinate-free, meaning that no knowledge is assumed of either the array geometry or the camera position. This allows for an ad-hoc placement of these elements. We also showed that, with aide from a face detector, a life-long learning approach can be taken to the problem of calibrating the regressor.

In this chapter we continue developing this line of coordinate-free methods. Now that a working localization methodology in place, we now propose a solution to the problem of audio tracking, namely TDOA tracking. A good TDOA tracking algorithm is necessary to compliment an accurate TDOA to pan tilt regressor. By leveraging simple assumptions about the human subjects creating the TDOAs, a tracking algorithm is able to give more accurate TDOA estimates than a single frame based TDOA estimation procedure.

One such assumption that we can exploit is that humans typically do not move too quickly or jump from one location to another in an instant. In particular, the TDOA estimate for a stationary sound source, a typical use-case scenario for TAC, should also be stable. These temporal assumptions can be integrated into a tracking model that makes for a more robust TDOA estimator, and as a result more accurate and responsive localization.

A naive approach to tracking could do something like median filtering of the TDOA vector estimates over time. In fact, this is what the original implementation of

the audio localization component to TAC was. However, integrating assumptions about the possible motion of the sound source being tracking can lead to a much more powerful TDOA estimation procedure.

In what follows we propose a particle filtering methodology for tracking the 21-D TDOA vector over sequential frames of audio. The methodology has three important innovations above the naive median filtering strategy outlined above. The first is that TDOAs from one frame to the next should not vary too much. This assumption should be explicitly integrated into any model of tracking. A second observation is that TDOAs can only occur from a feasible region of the 21-D space in which TDOA vectors lie. We propose a PD-Tree based model of this feasible region. It is well known that particle filters tend to break-down when the object being tracked have many dimensions to their state space. By modeling the feasible region, we alleviate this well known deficiency of particle filters by making the effective dimensionality of the TDOA space much lower.

The last contribution is a new particle weighting and resampling scheme inspired by results in online learning. The resampling scheme is such that we can leverage the PD-Tree model in a novel fashion that allows for averaging over different bandwidths in the tree. We will show in the experiments that this averaging scheme can improve over baseline schemes especially when a sound source enters regions that are not modeled well by a single global linear model. In addition, it is known that the weighting scheme used is much more robust to model misspecification than traditional particle filters.

The chapter opens by discussing traditional particle filters. It then discusses already existing coordinate-based methods for tracking audio sources, many of which utilize this traditional particle filtering approach. We then discuss the new particle filtering method based on the Normal Hedge online learning algorithm [CFH09]. In the experimental section of the chapter we give some results of our tracker and traditional particle filters on recordings from TAC with a moving speaker. Finally, we discuss the delay-and-sum beamforming used in TAC, which relies heavily on accurate TDOA estimates in every frame.

4.1 Particle Filters

Particle filtering is an approximation technique used to solve the Bayesian filtering problem for state space tracking [AMGC02]. More specifically, assume we have observations y_t and a state space x_t. Often the state space will consist of the position of the object of interest, and sometimes higher moments like velocity or acceleration. The goal of the particle filter is to keep a discrete set of particles which well-approximates the posterior density of the current state given the past observations $p(x_t|y_0,\ldots,y_t)$. In the TDOA tracking problem our observations y_t will be the PHAT correlations for a given frame of audio and the state space x_t will be composed of each of the $D = \binom{k}{2}$ time delays.

Often it is assumed that the state sequence follows a Markov process. Then, the dynamical system can be describe by the following two equations

$$x_t = g(x_{t-1}) + u_t \tag{4.1}$$

$$y_t = h(x_{t-1}) + v_t \tag{4.2}$$

where $g(\cdot)$ and $h(\cdot)$ are possibly nonlinear functions and u_t and v_t are noise contributions which need not be Gaussian. These equations describe the dynamics of the system and how observations are generated.

As in other works, we adopt the notation $x_{0:t} = \{x_0,\ldots,x_t\}$. The Bayesian filtering problem can then be posed as the following. Assuming that $p(x_{t-1}|y_{1:t-1})$ is known at time $t-1$, the posterior $p(x_t|y_{1:t})$ at time t can be estimated with,

$$p(x_t|y_{1:t}) \propto p(y_t|x_t)p(x_t|y_{1:t-1}) \tag{4.3}$$

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{4.4}$$

where $p(x_{t-1}|y_{1:t-1})$ is known as the *prior*, $p(x_t|x_{t-1})$ is the *transition* density and $p(y_t|x_t)$ is known as the *likelihood* function.

If we assume g and h to be linear and u_t and v_t to be Gaussian, then the optimal solution to the Bayesian filtering problem is the Kalman filter [Kal60]. Unfortunately for the audio localization problem, we have a mismatch with the Kalman assumptions on both fronts. First, the TDOA transition is known to be nonlinear as a function of

location as described in Chapter 3. Second, the noise on observations in the PHAT function are highly non Gaussian since they are controlled by such phenomenon such as reverberations and multipath reflections. As a solution to this problem, one popular way to approximate the Bayesian filtering problem in this nonlinear regime is through particle filters.

The *bootstrap* is one of the most popular particle filtering algorithms [GSS93]. Here, a weighting over m particles is chosen to approximate the posterior density. Let $w_t^{(i)}$ be the weight associated with particle i at time t. Then, a single iteration of the algorithm proceeds as follows:

1. **Sample:** draw m particles $x_{t-1}^{(i)}$ from the existing set of particles according to their weights $w_{t-1}^{(i)}$.
2. **Propagate:** Let the particles propagate according to the transition function, $x_t^{(i)} = g(x_{t-1}^{(i)}) + u_t$.
3. **Weight update:** Update weights according to $w_t^{(i)} = w_{t-1}^{(i)} p(y_t|x_t^{(i)})$ and normalize so they sum to one.

The result is a set of particles approximately distributed as the posterior density $p(x_t|y_{1:t})$. This sample set allows for computation of any quantity as a function of the posterior. For example, often we would like to estimate the mean of the posterior distribution which will be our prediction of the current state. This estimate is given by

$$\hat{x}_t = \int x_t p(x_t|y_{1:t}) dx_t \approx \sum_{i=1}^{m} w_t^{(i)} x_t^{(i)} \tag{4.5}$$

The weights are chosen to approximate the *relative* posterior density for their respective particles.

This popular variant of particle filters has been shown to perform well in the coordinate-based tracking literature [LJ07]. The key decisions for optimizing such a particle filtering algorithm are:

1. **Likelihood:** The choice of likelihood function, $p(y_t|x_t)$, is critical since this will govern how weights are calculated.

2. **Propagation function:** The propagation function $g(\cdot)$ is also essential and needs to be chosen accurately. In coordinate based methods g is chosen to be linear and u_k often to be Gaussian.

3. **Number of particles:** The total number of particles m. The larger m is the more computational load the system must undertake. Optimizing m is of paramount importance for real-time implementations.

More so than the other choices, the likelihood function is by far the most difficult. The true likelihood function for how PHAT observations are generated from a given sound source location seems very difficult to model. Nevertheless, it has been shown that some simple choices for the likelihood function can lead to good tracking performance [LJ07]. In making a choice for the likelihood function, first notice that we must have support over the entire observation space. If we don't meet this requirement, particles that occur with likelihood zero will get weight zero and die immediately. This is not the behavior we would like since particles that were performing well in the past may then suddenly die. Instead, we should want a more graceful way for particles to tend towards zero weight. As a result, often a mixture of a uniform prior over the entire observation space is mixed with the likelihood function to avoid this behavior.

One deficiency of the particle filter is that accurate tracking becomes very difficult when the state space becomes larger than a few positional locations (e.g. 2-D or 3-D locations). In TDOA tracking, the state spaces can potentially be much larger. For example, the seven microphones in TAC give rise to a 21-D TDOA vector space, but with arrays with more microphones the space can be even larger. The difficulty arises in the randomness need in u_k to generate enough variety of particles so that a few are close to a good state representation. One obvious remedy would be to increase the number of particles, but this causes the real-time feasibility of the algorithm to quickly diminish.

To alleviate this problem, when a coordinate system is known, then the state space can be represented as the 3-d position of the audio source. This makes the algorithm feasible with a small number of particles (typically < 100). In our coordinate-free approach, we take a similar dimensionality reduction technique by directly modeling the low dimensional structure on which the TDOAs lie via a PD-Tree. However, before introducing our algorithm we first discuss related work in coordinate based TDOA

Algorithm 4 Generic bootstrap based particle filtering audio tracking algorithm.

Initial Assumptions: At time t-1, we have the set of particles $x_{t-1}^{(i)}$ and weights $w_{t-1}^{(i)}$, $i \in \{1,\ldots,m\}$, being a discrete representation of the posterior $p(x_{t-1}|y_{1:t-1})$.

1: **Dynamics:** Propagate the particles through the transition equation $x_t^{(i)} = g(x_{t-1}^{(i)}, u_t)$.

2: **Weight Update:** Assign each particle a likelihood weight according to $w_t^{(i)} = p(y_t|x_t^{(i)})$. Then, normalize weights so that they sum to 1.

3: **Resample:** Resample m new particles from $\{x_t^{(i)}\}_{i=1}^m$ according to the weight distribution $\{w_t^{(i)}\}_{i=1}^m$. Let these be the new set of particles $\{x_t^{(i)}\}_{i=1}^m$ and assign uniform weight to each.

tracking.

4.2 Related Work

Particle filtering methods dominate the audio source tracking literature [LJ07, LS10, PKV08, TPC09]. The seminal work of Ward et. al is the first to popularize the use of particle filtering methods for audio tracking and is still widely regarded as state-of-the art [WLW03]. Further experiments and slight improvements on this method were presented in [LJ07]. This method is the focus of what follows realizing that the others mentioned above are all derived from this seminal work.

We reproduce the bootstrap particle filtering method for audio source tracking in Algorithm 4. The predicted state at each step of this algorithm is the weighted mean $\hat{x}_t = \sum_{i=1}^m w_t^{(i)} x_t^{(i)}$. Here the state space is chosen to be 3-d Cartesian coordinates $x_t^{(i)} = [p_x p_y p_z]$ and the dynamics g is chosen to be the identity with spherical Gaussian noise for u_k. The size of the Gaussian noise u_k is a tunable parameter that must coincide with the assumptions about how quickly the objects being tracked can move.

The major choice in the algorithm is how to perform the weight update step, in particular, what choice should be made for the likelihood function $p(y_t|x_t^{(i)})$. The choices for this function can arise either from GCC based methods or steered beamforming based methods. For example, a simple steered beamforming based approach is as follows. For the weight update in Algorithm 4, let $p(y_t|x_t^{(i)}) = F(y_t, \Delta(x_t^{(i)})$ where F

calculates the steered response power of the current frame of audio steered towards $x_t^{(i)}$.

More computationally efficient methods for representing the likelihood function were presented in [WLW03] based on PHAT transforms. The idea for the likelihood here is to define a function that combines how close the current particle is to the largest peaks in the PHAT correlation from each pair of microphones $p \in \{1,\ldots,D\}$. This will be the method we use in the work presented in this chapter. In particular we use the following.

First, to identify the peaks in a given pair's PHAT function we take a simple z-scoring method. Let $[A]_+ = max(0,A)$. Then, for each PHAT correlation R_p let it undergo a z-scoring transform as follows (note from here on we drop the subscript t for ease of notation):

$$Z_p(\tau) = \left[\frac{R_p(\tau) - \mu_p}{\sigma_p} - C\right]_+ \tag{4.6}$$

where μ_p, σ_p are the mean and standard deviation of R_p over a fixed bounded range of τ, and C is a constant requiring that peaks be at least C standard deviations above the mean. This performs well to find a small, fixed number, of peaks K_p in each R_p.

We now define $p(y|x^{(i)})$ in terms of these peaks:

$$p(y|x^{(i)}) \propto p_0 + \sum_{p=1}^{D}\sum_{l=1}^{K_p} Z_p(\tau_l)\mathbb{N}(\tau_l;\Delta(x^{(i)})_p,\sigma_z^2) \tag{4.7}$$

where $\Delta(x^{(i)})_p$ the TDOA associated with pair p derived from the 3-D location $x^{(i)}$, $\mathbb{N}(x;\mu,\sigma^2)$ is the density under a normal distribution evaluated at x with mean μ and variance σ^2, and Z_p has K_p non-zero entries each of which are at τ_l. The parameter p_0 is the background likelihood that determines how much likelihood is given to any TDOA regardless of the observation. This parameter is essential for this kind of particle filter so that the likelihood function never evaluates to 0. Otherwise a particle's weight can never abruptly vanish. The variance parameter σ_z^2 controls how much weighting is given relative to how far each state is from the peaks in the corresponding PHAT series. So, a particle will be given high likelihood if the particle's derived TDOA matches well with the largest peaks in the observed PHAT series. Conversely, if the derived TDOA is far from any of the observed peaks it will be given a very low likelihood.

A nice property of this choice of likelihood is that it does not rely solely on the maximum of each PHAT series being accurate (a similar advantage was observed

between steered beamformers over the 2-step localization procedure discussed in Chapter 3). Since often the peaks in the PHAT localization are corrupted due to reverberations or multipath reflections, relying heavily on only these maximum peaks is not robust. The likelihood defined in Equation 4.7 neither relies too heavily on the accuracy of a single pair of microphones, nor on the largest peak in each pair's PHAT series. Secondary peaks can contribute substantially to the likelihood as well. As we will see, integrating a particle filtering based tracking method into the localizer will lead to a much stabler and robust localization method.

4.3 Normal Hedge Based Particle Filter

In this section we introduce the Normal Hedge based particle filter. This particle filter, although very similar to the traditional particle filter introduced above, will have several advantages. First, the resampling scheme will not require particles to be resampled every iteration. In fact, particles will remain "alive" for as long as they perform well. Secondly, the requirements of the algorithm will allow for much more flexibility in specifying a likelihood function. Recall that in Equation 4.7 we had to define a parameter for the background likelihood p_0, otherwise particles could quickly go to zero weight and die. No such requirement is needed by the particle filter presented here, moreover, the guarantee that will be given is relative to the defined likelihood function. This means that the resulting Normal Hedge particle filtering algorithm will perform well as long as the likelihood function encourages good tracking performance (i.e. high likelihood scores indicate that the particle matches the observation well).

Before introducing the full Normal Hedge particle filter we first discuss the Normal Hedge online algorithm for predicting from a group of experts' advice, initially presented in [CFH09].

Normal Hedge

The Normal Hedge algorithm is a parameter-free online algorithm for hedging over the predictions from a group of N experts [CFH09]. One of the barriers to practical implementations of previous online learning algorithms was that they all contained

Algorithm 5 Normal Hedge parameter-free online learning algorithm.

Initial Assumptions: At time $t-1$ we're given the cumulative loss of each expert $R_{t-1}^{(i)}$ and the discrete weighting $w_t^{(i)}$. Initially $R_0^{(i)} = 0$ and $w_1^{(i)} = 1/N$ for all i.

1: **Update Losses:** Each action incurs a loss $\ell_t^{(i)}$ and the learner incurs loss $\ell_t^A = \sum_{i=1}^N w_t^{(i)} \ell_t^{(i)}$.

2: **Update Regrets:** Update the cumulative regrets $R_t^{(i)} = R_{t-1}^{(i)} + (\ell_t^A - \ell_t^{(i)})$

3: **Update Weights:** First, find $c_t > 0$ that satisfies $\frac{1}{N}\sum_{i=1}^N \exp\left(\frac{([R_t^{(i)}]_+)^2}{2c_t}\right) = e$. Then, update weight distribution for round $t+1$ by $w_{t+1}^{(i)} = \frac{[R_t^{(i)}]_+}{c_t} \exp\left(\frac{([R_t^{(i)}]_+)^2}{2c_t}\right)$. Normalize the weights so they sum to one.

a learning parameter that was very important to tune correctly for good performance. Normal Hedge has no such parameter, yet still has a very strong performance guarantee like that of the previous online algorithms.

The setup for the algorithm is as follows. At each iteration t expert i makes a prediction that has an associated loss $\ell_t^{(i)} \in [0,1]$. The notion of loss in this setting is very general, but in most cases is typically derived as a function of the expert's prediction and the actual observation (e.g. the difference between the prediction and the observation normalized to the $[0,1]$ range). The algorithm maintains a discrete probability distribution over the experts $w_t^{(i)}$. After observing the losses, the learner itself incurs a loss according to the expected loss under this discrete distribution,

$$\ell_t^A = \sum_{i=1}^N w_t^{(i)} \ell_t^{(i)} \tag{4.8}$$

The notion of *regret* is the essential quantity of interest in online learning. The algorithm's *instantaneous regret* is defined as $r_t^{(i)} = \ell_t^A - \ell_t^{(i)}$ and the *cumulative regret* up to time t is defined as

$$R_t^{(i)} = \sum_{\tau=1}^t r_\tau^{(i)} \tag{4.9}$$

Intuitively the cumulative regret measures how well the algorithm is doing relative to a single action chosen to predict at all previous iterations up to t. The goal for an online algorithm is to minimize the cumulative regret of the algorithm relative to any given expert (in particular, the best expert in hindsight).

The Normal Hedge algorithm is given in Algorithm 5. It requires no parameters and the computational needs are also simple. The algorithm must maintain the weights and regrets over each of the N experts and also a line search is needed to solve for c_t in the weight update stage.

The guarantee proved in [CFH09] is that the cumulative regret to the best ε percentile of experts will be small. In particular at time t the cumulative regret of Normal Hedge to the ε percentile expert will be $\mathbb{O}(\sqrt{t(1+\ln 1/\varepsilon)}+\ln^2 N)$. This is more general than the regret bounds that already existed in the online learning literature which only considered regret to the "best" expert in hindsight. The notion of "ε percentile" is a more useful bound in the sense that in many practical situations there are many experts among the N which are almost as good as each other. As a result, guaranteeing performance relative to the absolute best is often too strong. Moreover, the bound given in [CFH09] is still competitive with other known results when considering the "best" expert case by setting $\varepsilon = 1/N$.

NH-pf Derivation

Transforming the Normal Hedge algorithm into a particle filtering algorithm is quite natural. We must only transform the terminology "experts" into "particles" and we're most of the way towards a Normal Hedge based particle filtering algorithm. A recent paper was published that was that first to describe how the Normal Hedge algorithm can be used as a particle filter [CFH10].

In the tracking problem we consider an expert to be a predictor of a sequence of hidden states $(x_1,\ldots,x_t)$ up to time t. This sequence of states is a proposed explanation for the sequence of observations $(y_1,\ldots,y_t)$. Instead of a likelihood function $p(y_t|x_t)$ like in particle filters, for the Normal Hedge tracking algorithm we must define a loss function on which to measure each expert's performance. The loss $\ell_t^{(i)}$ for expert i should measure how well an experts sequence of states matches the sequence of observations.

After defining this loss, we nearly have all the components needed to utilize Normal Hedge in the tracking framework. However, there is a computational issue at hand, namely, the exponential explosion in possibilities for state space sequences. Imagine we could run Normal Hedge over this enormous number of experts. Luckily,

Algorithm 6 Normal Hedge based particle filter.

Initial Assumptions: At time t-1, we have the set of particles $x_{t-1}^{(i)}$ and Normal Hedge weights $w_{t-1}^{(i)}$, $i \in \{1,\ldots,m\}$

1: **Regret Update:** Obtain losses $\ell_t^{(i)}$ for each particle and update *discounted cumulative regrets* $R_t^{(i)}$ (Equation 4.10).

2: **Resample:** For each particle $x_t^{(i)}$ with $R_t^{(i)} < 0$, resample a new particle in its place

   1. Choose a current particle $x_t^{(k)}$ according to $\{w_t^{(i)}\}_{i=1}^m$.

   2. Let the new particle $x_t^{(i)} = x_t^{(k)} + u_k$, where u_k is Gaussian noise. **(Coordinate-free only):** Project back onto the TDOA manifold using the PD-tree projection.

   3. Assign $R_t^{(i)} = (1-\alpha)R_t^{(k)} + (\ell_t^A - \ell(x_t^{(i)}))$.

3: **Weight Updates:** Update the weights of each particle according to the Normal Hedge procedure (see Algorithm 5). Normalize them to one.

4: **Dynamics:** Propagate the particles through the transition equation $x_t^{(i)} = g(x_{t-1}^{(i)})$.

we'd have one advantage on our side because the Normal Hedge weighting would give many experts weight zero except for a core group that are outperforming the predictions of the algorithm itself. Nevertheless, some approximation is necessary, but this sparsity property will ease the requirements of any approximation. The approach we take will sample from this large set of experts in a very similar fashion to that of the bootstrap particle filter described earlier in this chapter.

Just as the particles in a particle filter are a discrete approximation to the posterior density, we will utilize a set of particles to approximate the induced distribution by Normal Hedge over the set of state sequences. The Normal Hedge algorithm for TDOA tracking is given in Algorithm 6. Notice that a further simplification is taken to the problem by only maintaining the *discounted cumulative regret*

$$R_t^{(i)} = (1-\alpha)R_{t-1}^{(i)} + (\ell_t^A - \ell_t^{(i)}) \tag{4.10}$$

where the parameter α controls how much memory our tracking algorithm should have in terms of penalizing losses observed in the past. This approximates the need for track-

ing sequences of states. Since typically we only want to predict the current state, this is an acceptable simplification. The weights can then be calculated according to the steps involved for computing $w_t^{(i)}$ in the Normal Hedge algorithm (see Algorithm 5). What remains as a critical algorithmic choice is how we compute the loss $\ell_t^{(i)}$ for each particle, analogous to the decision of the likelihood function in particle filters.

A nice property of this filter (NH-pf) is that the resampling procedure for particles emerges naturally from the Normal Hedge weighting function. A particle will be given zero weight whenever its cumulative regret has started to perform worse than the algorithm itself, and at this moment it is resampled near a particle that has good historical performance.

Another nice property of NH-pf is that it can still maintain good tracking performance when the loss function has a modeling mismatch with the true observation process [CFH10]. Chaudhuri et. al show that if a traditional particle filter has a likelihood function that mismatches the true underlying process, then its performance will break down much quicker than the corresponding NH-pf. This final observation could prove to be advantageous in the TDOA tracking scenario. As stated earlier, choosing a likelihood function for TDOA tracking is somewhat arbitrary since the process for generating a PHAT observation from a given source location is extremely difficult to model. For this tracking problem and many other of practical importance, a model mismatch of this kind is unavoidable.

TDOA Tracking with Coordinate-Free NH-pf

We now describe how we track TDOAs in a coordinate-free fashion. First, we expand the state space to be that of the entire TDOA vector (for TAC a 21-dimensional state space). In addition, we must specify what loss function we will be using to calculate regrets relative to. We will utilize the negative likelihood function described in Equation 4.7

$$\ell_t^{(i)}(x_t, y_t) = -p(y_t|x_t^{(i)}) \tag{4.11}$$

As discussed earlier, tracking in this many dimensions becomes difficult, but we also know that our TDOAs lie on a low dimensional manifold. In Chapters 2 and 3 we discussed PD-Tree based models of the TDOA structure. We again utilize this to

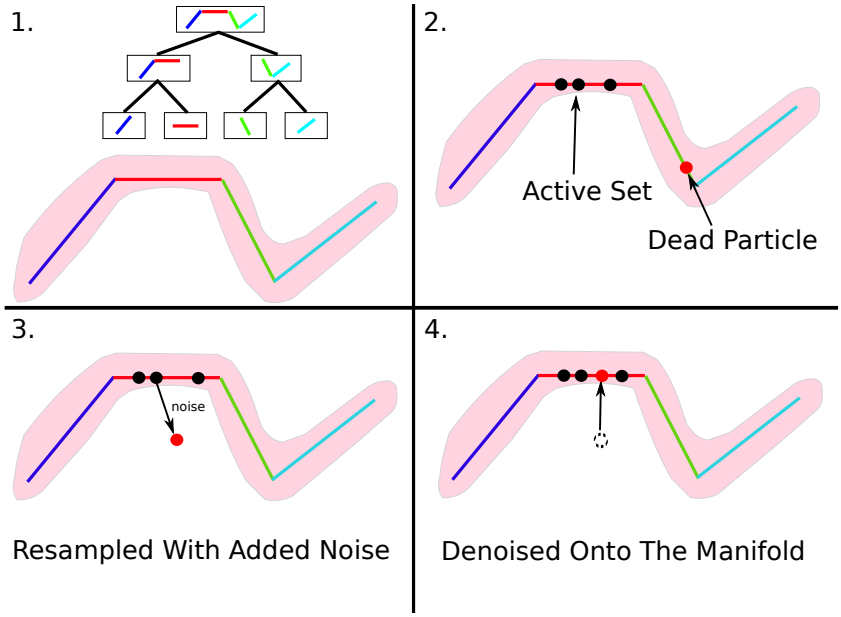


Figure 4.1: Depiction of a dead particle resampled and projected back onto the manifold.

alleviate this high dimensional tracking problem. During the resampling step for new particles, after noise is added to the newly born particle it is projected back onto the TDOA structure via the PD-Tree model. This process is depicted in Figure 4.1.

First, the corresponding leaf node for the newly born particle is found. Then, the particle is denoised by projecting it onto the principal components of local piece of the manifold stored within. The overall effect is a tracking algorithm that is constrained to have a state-space lie on the low dimensional structure captured by the PD-Tree.

Note that there is a bias variance trade-off as you descend the PD-Tree in terms of the principle components models stored in each node. The nodes higher up in the tree have small variance since the bulk of the training data was used to learn the principal directions. However, the bias is also large at these nodes since a linear model is not appropriate at this granularity. As you descend the tree the variance increases, whereas the bias decreases as you approach locally linear regions.

Moreover, the fit from individual nodes may vary across even their own partition region. Consider a given internal node of the PD-Tree. For a certain region of the cell's partition the linear model may be a good fit, whereas in other regions of the cell's partition it may be poor. It is clear from this argument that the best node that fits the

true TDOA for a given source location will vary with location both across the tree and possibly in depth as well.

It then makes sense to consider using the entire PD-Tree during the projection step instead of just the leaves at a fixed depth. A natural way to accomplish this emerges from the NH-pf resampling scheme by making a slight alteration to the projection step with the PD-Tree discussed above. After resampling a newly born particle and before projecting it back onto the manifold via the PD-Tree, first pick a depth uniformly at random in the PD-Tree. Then, traverse the PD-Tree with this newly born particle to this depth in the PD-Tree and use the principal components stored in this node.

This random strategy will have the nice property that it will naively find the correct model depth for the current sound source location over time. Depths that are chosen that are poor models will have particles die soon thereafter, whereas particles that are drawn from depths that perform well will survive. We will examine this procedure in the experiments that follow.

4.4 Experiment: Moving Speaker

The experiments that follow were conducted from recordings of real speakers talking and moving slowly while facing the array. We describe each individual experiment in detail in what follows.

Setup

To build a PD-tree we first collected a training set of TDOA vectors from our microphone array. We accomplished this by moving a white noise producing sound source around the room near typical locations that sitting or standing people would be interacting with the display. This resulted in approximately 20,000 training TDOA vectors to which we built a PD-tree of depth 2. In each node of the PD-tree we store the mean of the training data and the top $k = 3$ principal directions.

Here are the parameter settings we use for the experiments that follow. We use $m = 50$ particles for each type of particle filter examined. The discounting factor for NH is set to $\alpha = 0.05$.

We made several real audio recordings of a person walking throughout the room facing the array and talking. We describe each experiment in detail in what follows.

Usage of Manifold Modeling

This first experiment has a person walking and counting aloud while facing the array. The person's path goes through the center of the room far from each microphone. Since TDOAs evolve more slowly when the sound source is far from each microphone we'd expect this to be well modeled by the root PCA of our PD-tree. We compare using the root PCA versus no projection step at all for both a standard particle filter (PF) and the Normal-Hedge particle filter (NH).

Figure 4.2 depicts such a comparison. Here we show tracking results from two microphone pairs that are typical of the remaining pairs (i.e. two coordinates of the 21-D TDOA state). In green is shown Z_t^p where its magnitude is represented by the size of the circle marker. The sound source moved in a continuous and slowly moving path so we'd expect each TDOA coordinate to follow a continuous and slowly changing path as well. The trackers with the PCA projection step outperform their counterparts without the projection.

From this single trial run, NH-pca seems to have a slight advantage over PF-pca from time to time, but the two algorithms are competitive in performance. However, a more closer examination shows an advantage to NH-pf. When averaged over 25 independent runs over this audio recording the NH-pf with pca is slightly more accurate and clearly stabler than the standard PF. Figure 4.3 depicts the RMSE of each tracker averaged over 25 independent trials. The RMSE is calculated coordinate-wise relative to the maximum of the PHAT series for each frame. Since the maximum derived TDOA is often accurate, but sometimes widely inaccurate (especially during periods of silence), we smoothed each RMSE series using an exponential moving average with $\alpha = 0.05$. It is clear from these plots that the pca based methods are outperforming the non-pca ones, and the NH methods have an advantage over the standard PF methods.

Figure 4.4 depicts the variance of each method as a function of time. For each time t, first the norm of the state vector was calculated, and then we computed the variance of this norm across the 25 runs. The variance increases with time for all trackers

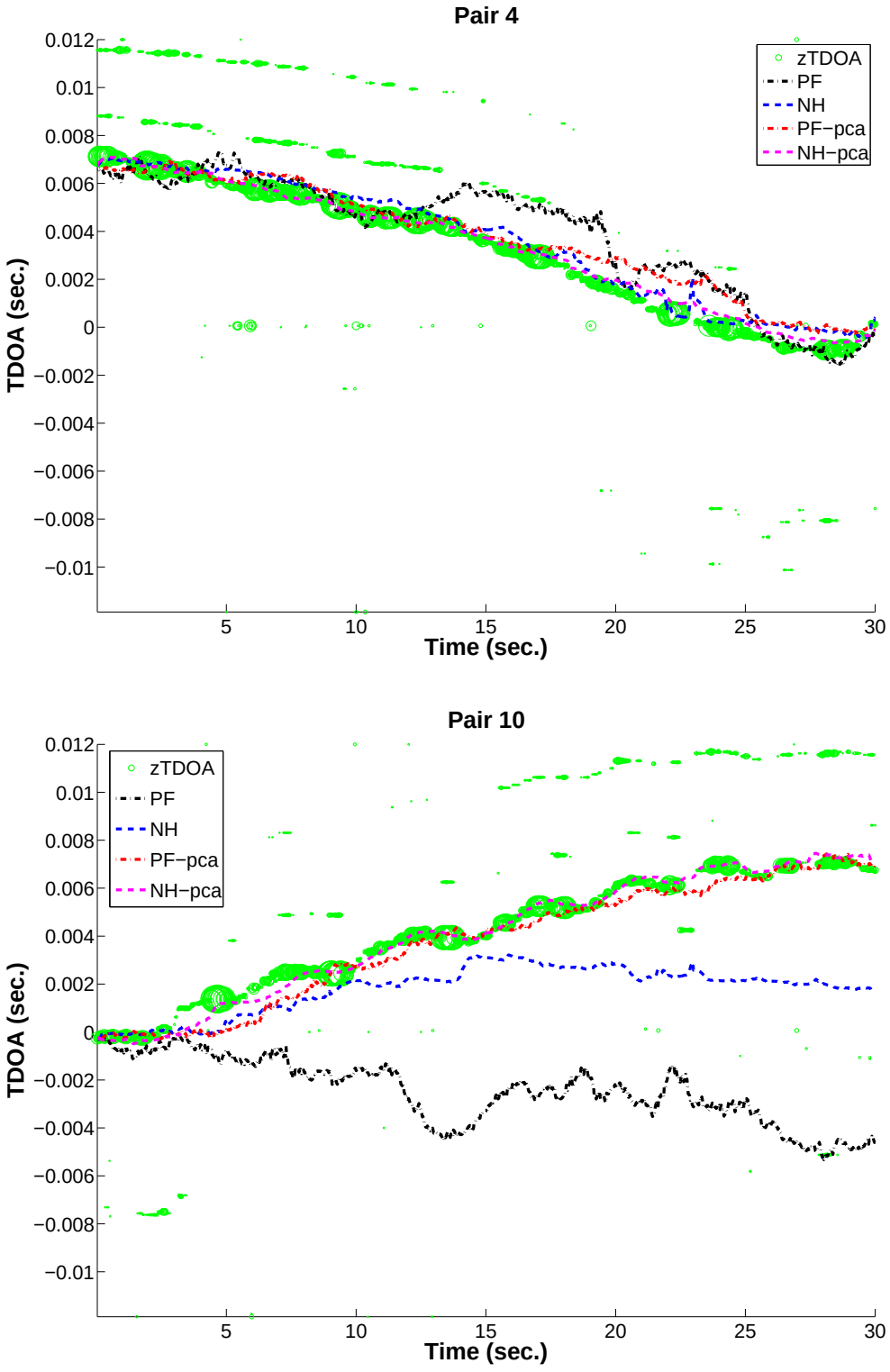


Figure 4.2: Performance of NH and PF with and without using a global PCA projection for denoising.

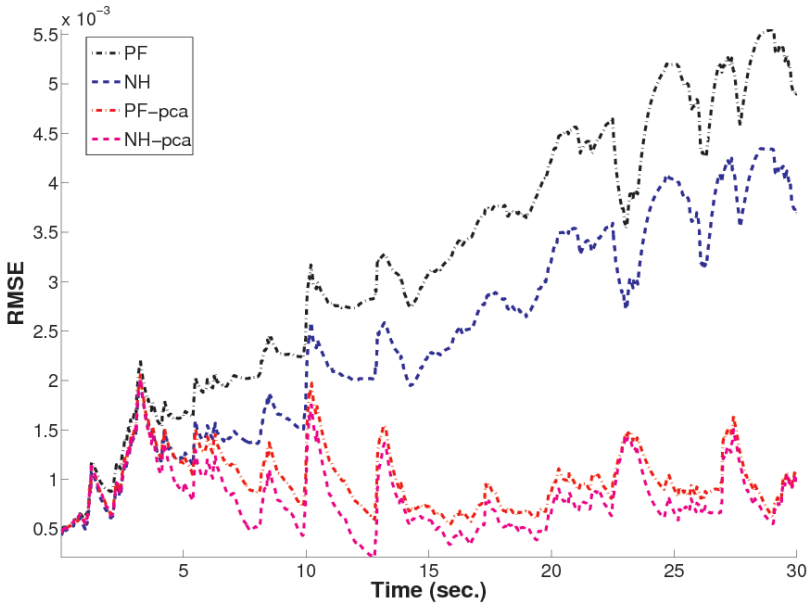


Figure 4.3: RMSE over 25 independent runs of each of the trackers.

except that of NH-pf with the pca projection. When comparing NH to PF, the NH trackers have much less variance than their PF counterpart. It's clear from these results that the NH-pf with the pca is a very stable and accurate tracker.

Remember that there are only 50 particles to track a state that is 21 dimensional. There are no dynamics involved in our particle filters, so the resampling stage alone has to include enough randomness for the source to be tracked as it moves. When the manifold model is not used the amount of randomness needed is too large for 50 particles to be able to track on all D dimensions. However, when a model of the manifold is used, effective tracking results can be had. Moreover, it should be noted that the NH version uses less randomness since it only resamples when the weight of a particle becomes zero. Despite this, the NH versions are able to have a competitive performance with standard particle filters.

Testing Different Manifold Models

The setup of this experiment is exactly the same as the last except the path the speaker took traveled much closer to some pairs of microphones at certain points in

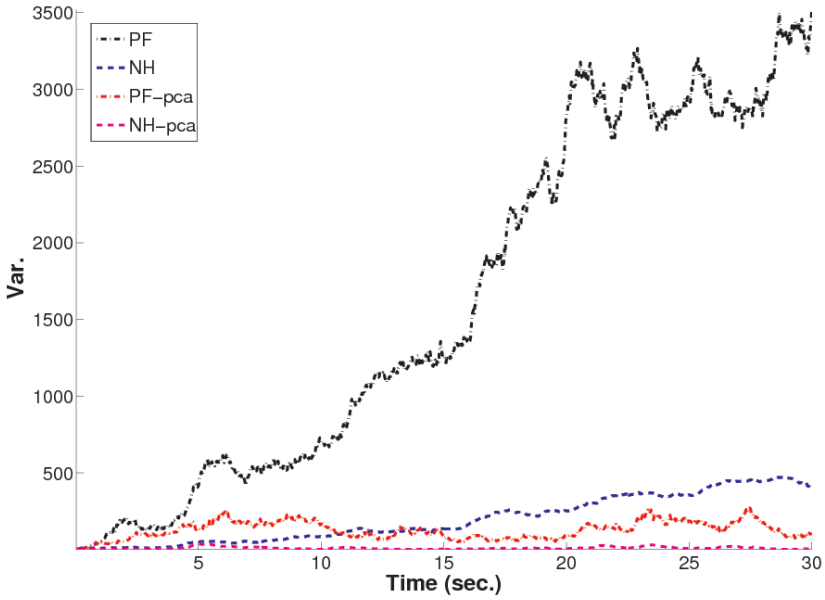


Figure 4.4: Variance over 25 independent runs of each of the trackers.

time. When a sound source is moving close to some set of microphones, the TDOAs involved with those microphones will change much more rapidly and non-linearly. With this path we hope to examine the usefulness of deeper nodes in the PD-tree. We will test the PD-Tree using only the nodes at a fixed depth ($d = 0, 1, 2$) and also the randomized scheme for choosing uniformly among the depths (see discussion in previous section for a full description of the randomized strategy).

Since the performance of NH was superior when using the global PCA projection we only examine NH in this experiment. This will allows us to explore the randomized manifold modeling scheme. In a standard particle filter, no benefit is gained by adopting this randomized strategy since all particles are resampled at each iteration. Thus, the random strategy in a standard filter can never will allow particles to "gravitate" towards the correct depth.

Figure 4.5 is a similar figure to that discussed in the previous section. The particle filtering variants examined here use projections at fixed depth zero (NH-0), one (NH-1), and two (NH-2). The random strategy is also examined (NH-rand). It is clear that somewhere between 50s-70s the location of the sound source is modeled poorly by

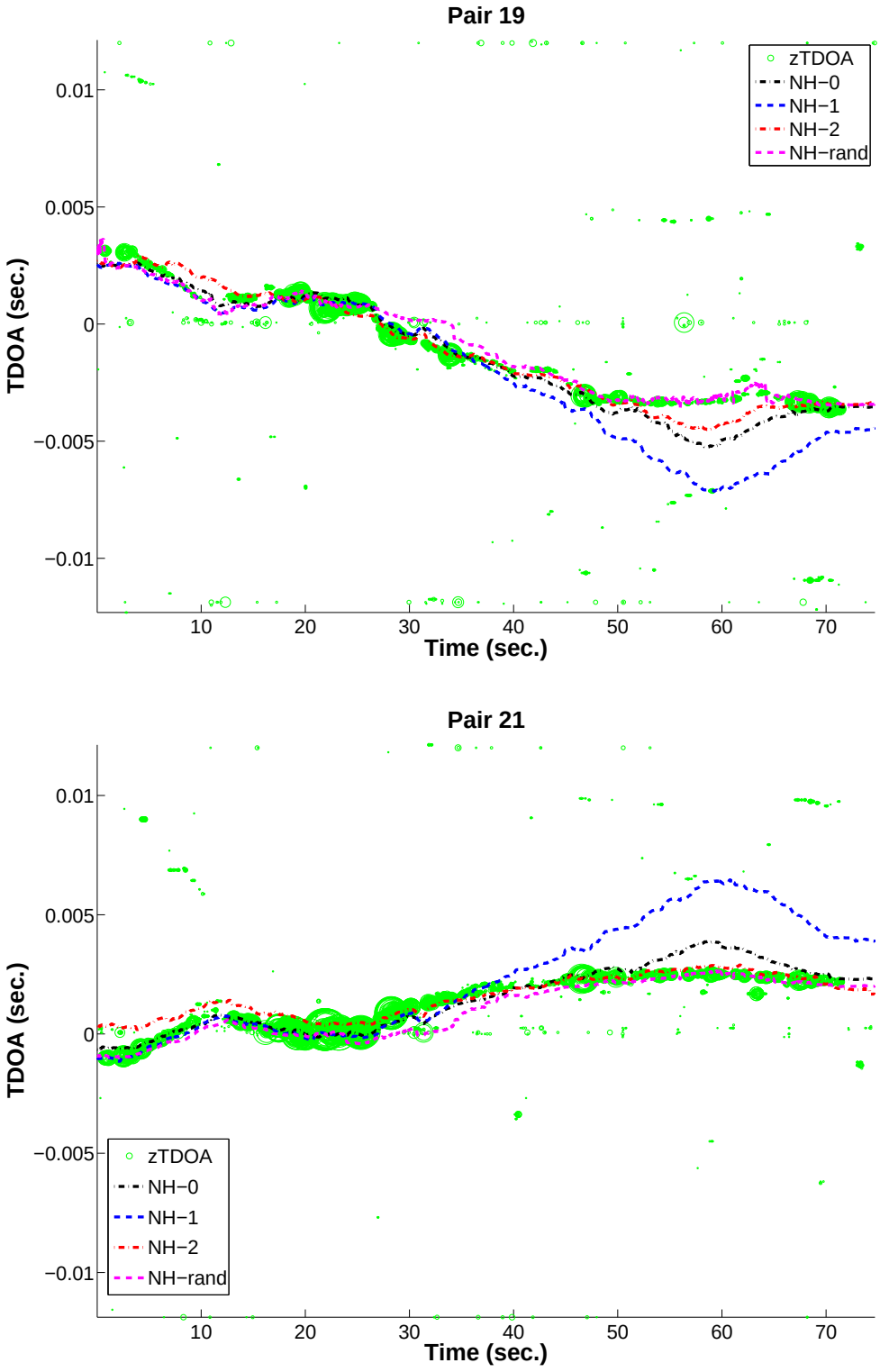


Figure 4.5: Using various depths in the PD-tree as part of the projection step.

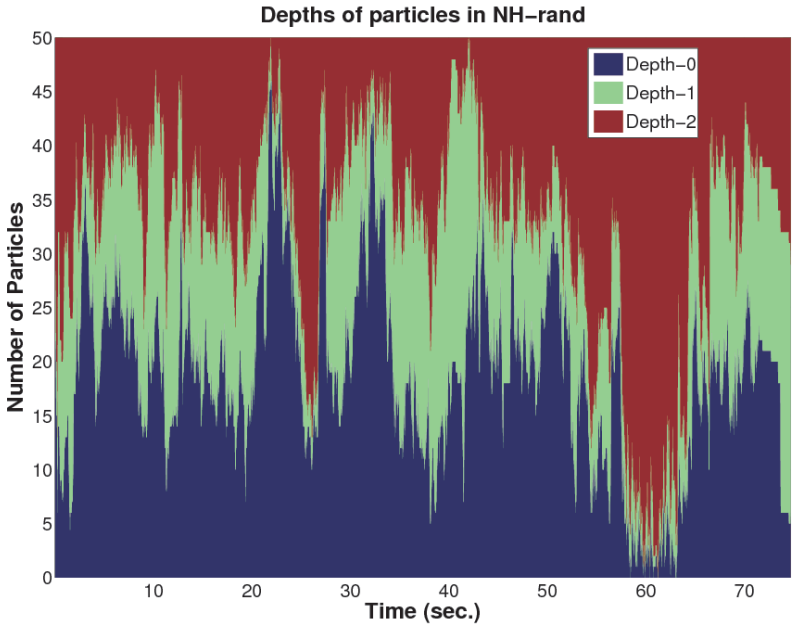


Figure 4.6: For NH-rand, the PD-tree depths at time t that the m particles have been sampled from last.

the global PCA at the root and is better modeled by the PCA at level 2. However, it is only for this short duration where this modeling transition takes place. Depth's 0 and 1 performed particularly poorly in this region, while depth 2 has a significant advantage.

However, the best performing tracker was one that utilized the entire tree structure in a random fashion. By allowing particles that died to birth at a random depth, there was a clear pressure for particles to transition from a depth-0 model to a depth-2 model rather quickly by NH-rand. This can be seen in Figure 4.6. Here we depict what proportion of the 50 particles at time t were last sampled from which depth by a stacked bar graph. Nearly all the particles during this time period that were sampled from depth-2 are staying alive during this period. This is a rather intuitive result since a particular node's PCA model may only be good for tracking in a small region of the entire 21 dimensional space that its PD-tree node represents. When the sound source exits this region, some other depth in the tree may become a better model. Using the randomness over time by NH-rand naturally captures such transitions.

Figure 4.7 shows another recording of a sound source moving at constant speed

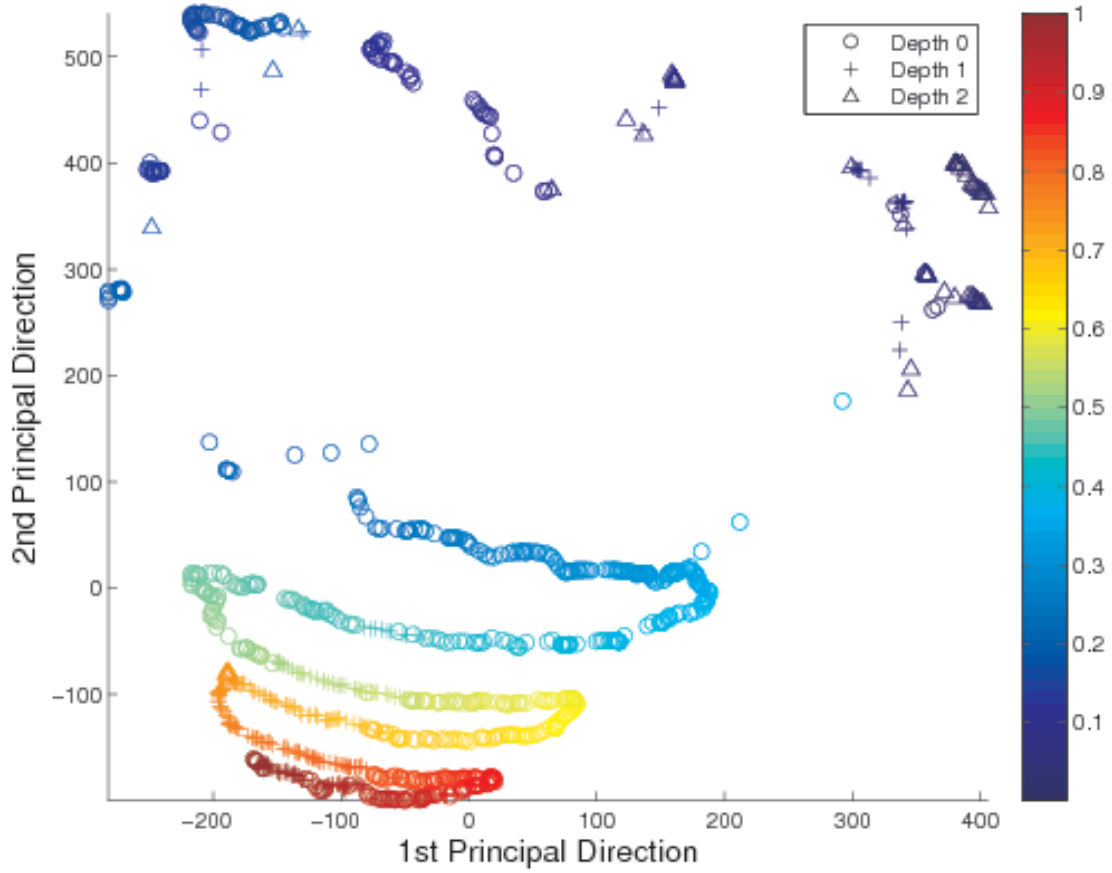


Figure 4.7: Sweeping path for NH-rand on top 2 principal directions of root PCA.

in a back-and-forth sweeping path. Each sweep starts beyond one end of the interactive display and continues across the opposite end. This is repeated at various distances away from the display. The TDOA vectors predicted by NH-rand are projected on the top 2 principal components of the root PCA. Colors indicate time, dark blue being the earliest part of the path that started approximately 1m from the display and red is the last segment of the path approximately 12m away. The change in TDOAs is greatest when near the microphones on the display resulting in a wide spacing of points. The markers indicate which of the 3 depths the majority of the NH-rand particles were last sampled from. In the center of the room the root-PCA performs best, whereas near the display on the right side depth 2 dominates, and far from the display depth 1 is best.

4.5 Delay-and-Sum Beamforming

The implementation of an effective tracking methodology gives more accurate and stable TDOA estimates from frame to frame. As mentioned before, this allows for an improved localization system. There are also other benefits of improved accuracy in TDOAs. Namely, it also allows for a simple beamforming scheme to be effective.

Possibly the simplest of beamforming techniques is the delay-and-sum (DEAS) beamformer. The idea is to first appropriately delay the signals from each microphone so that the desired speech is aligned, and then sum them together to enhance this speech and suppress sounds from other locations. Let $x_i(t)$ be the time-domain signal for microphone i. Given the TDOA vector Δ_{ij} we can determine these appropriate delays and sum according to $b(t) = x_1(t) + \sum_{i=2}^{k} x_i(t - \Delta_{1i})$.

After appropriate alignment of the desired sound, other sounds arriving from different locations are often misaligned and decorrelated, meaning that when summed they tend towards zero. Moreover, the aligned speech is averaged giving a smaller variance in its response. A more detailed examination of DEAS beamforming can be found in [JD01].

A stable TDOA vector with respect to time is essential for DEAS beamformers. If TDOAs change rapidly with respect to time, the audio can appear very poor and choppy. With the NH-rand particle filter the TDOA is both stable with respect to time and follows closely the desired true TDOA. Thus, we are able to beamform to the user's speech effectively from frame to frame. A DEAS method is implemented in TAC, and the beamformed audio is used as the audio for the recorded video whenever a user initiates a recording.

An example of a recording from a single microphone and that of the DEAS beamformer is given in Figure 4.8. The example here is a real recording of a speaker counting after the tracker has converged on their location. A single channel is shown in blue, while the DEAS output is shown in red. During periods of silence between the spoken numbers it is clear that the background noise is better suppressed than on the single channel. In fact, the SNR improvement of the beamformer over the SNR of a single channel is 7.37 dB. More complete SNR results are presented in Table 4.1.

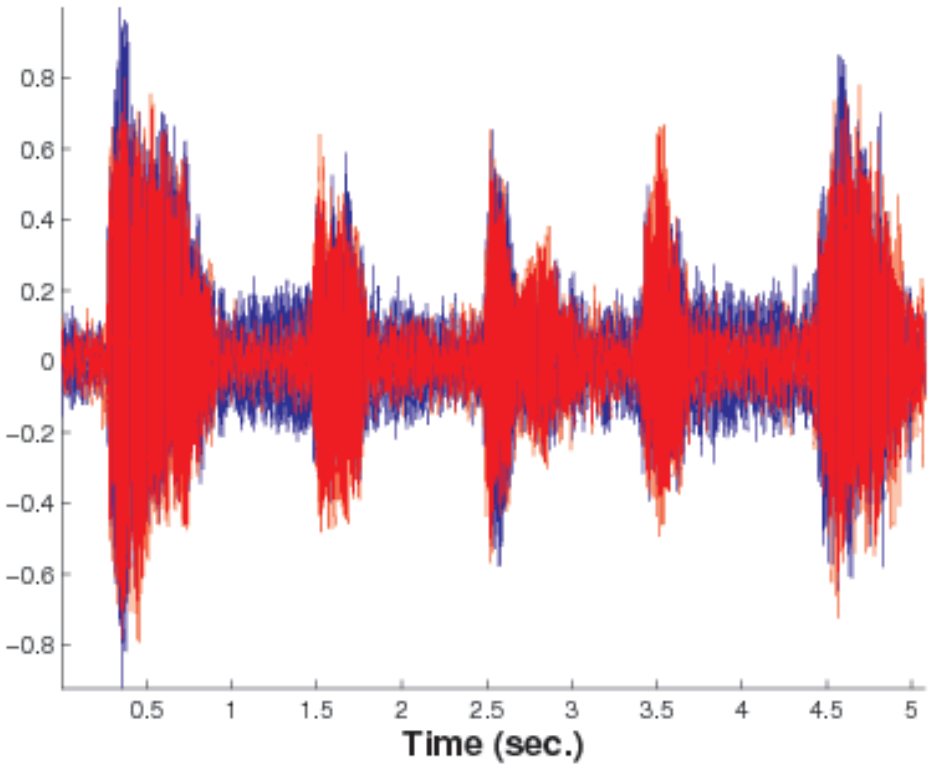


Figure 4.8: Overlay of a single channel (blue) with the output of the beamformer (red) on a signal of a speaker counting.

Table 4.1: SNR for DEAS beamformer and a single channel from TAC recordings.

Channel 1	**DEAS**	**Improvement**
6.68 dB	14.05 dB	7.37 dB

Acknowledgements

This chapter is based on unpublished work that is currently in submission as of the writing of this thesis. It is joint work with Yoav Freund. The dissertation author is the primary investigator and author of this work.

Chapter 5

Hardware Solutions

As we observed in the discussion of beamforming in Chapter 4, the analog microphones used in TAC are fraught with noise interference. This corruption can be viewed clearly especially when no signal is present: we would expect a flat signal, but instead we see interference (see single channel from Figure 4.8). Although there is noise from background elements such as the air conditioning and other ambient noise, the primary cause of this interference is from distortions occurring during the transmission of the analog signal from each microphone to the digitizer. In this section we study an implementation on an FPGA of the microphone capture component of the localization system in TAC. The system we present includes the use of microelectromechanical systems (MEMS) microphones which digitize the audio signal on-chip.

These MEMS microphones are very small (2-3mm) and are typically used in small consumer electronic devices such as laptops and cell phones. An FPGA processing the audio signal from these microphones are perfect for use in TAC for a variety of reasons. First, since these microphones digitize the audio signal on-chip, the transmission of the signal is very robust to interference from other nearby electronics. This is why MEMS microphones are excellent in electronic devices. Since they digitize the signal on-chip, the nearby other electrical components do not interfere as the digital signal is being transported across the device (e.g. along wires from the microphone to the audio codec on the mother board). Analog microphones, the type TAC currently uses, are fraught with this kind of interference.

Another reason to consider such an FPGA based device is that it allows for

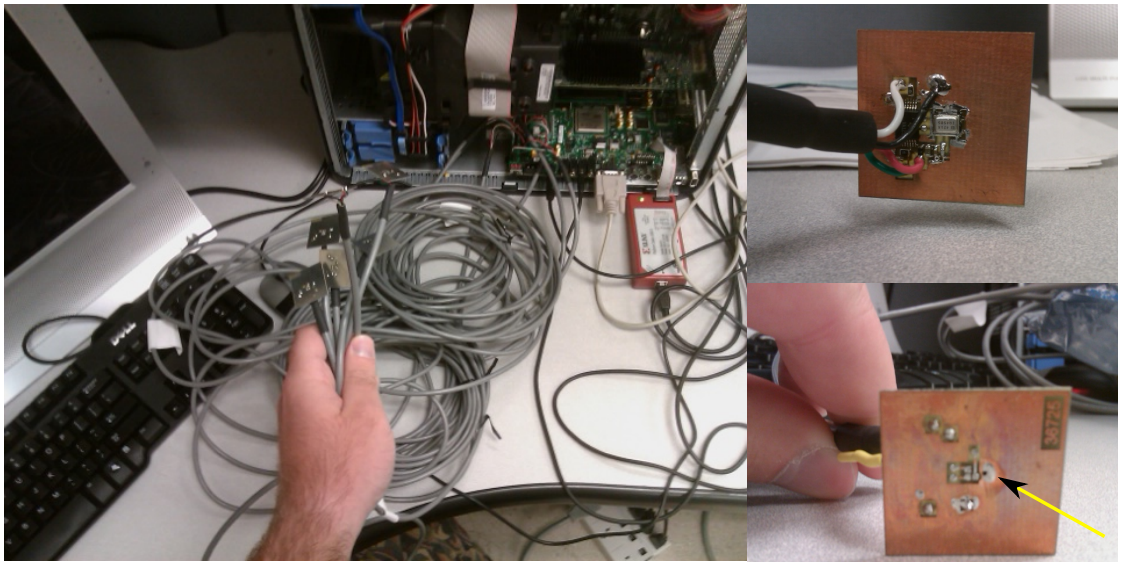


Figure 5.1: FPGA and MEMS microphones. FPGA is attached via PCIe slot of a Linux workstation with 6 MEMS microphones (left). Back (upper right) and front (lower right) view of a single MEMS microphone. The pressure sensitive hole for the microphone is labeled with a yellow arrow.

cheaply deploying many such setups. As mentioned earlier, to localize well across a large area, the microphone array needs to have a large aperture (i.e. microphone elements should be placed far apart). The FPGA setup will act as a microphone capture system, including digitization of the signal, allowing for several replicas of the setup to be deployed across a room. The cost of such a device is also much improved since we can utilize low-end FPGAs. The MEMS microphones are about $2 each and each board is about $300. This is a much improved cost compared to the current TAC setup which has a $2000 digitizer and a $100 cost for each analog microphone.

A special thanks must be extended to Mr. Thomas Raymond at Qualcomm Research for helping us acquire these MEMS microphones and mounting them on a board that could drive the digitized signal across a 15' line. This allows each microphone to be placed up to 15' away from the board itself (see Figure 5.1 for a picture of the FPGA and up-close MEMS microphone).

The rest of the chapter proceeds as follows. We first briefly introduce FPGAs and MEMS microphones. We then discuss the design and engineering challenges that

needed to be overcome in order to capture the MEMS microphone audio and process it on the FPGA board. We proceed by describing an experiment of the frequency sensitivity of the MEMS microphone compared to that of a typical analog microphone.

5.1 FPGAs and MEMS

A field-programmable gate array (FPGA) is an integrated circuit consisting of user programmable logic cells. Each cell can be configured by the user to do a variety of simple functions (e.g. simple boolean logic of two input signals). Multiple cells can be organized together to do tasks of varying complexity ranging from a simple MUX to a complete microprocessor. The user programs an FPGA using a hardware description language (HDL). We will utilize a Xilinx Vertex 5 series FPGA in the discussion that follows and most of the logic was programmed with verilog, an HDL language.

FPGAs can often implement a task more efficiently and consuming less power than by programming the same task in software on a general purpose CPU. Typical uses include ASIC design prototyping and computer hardware emulation, but there has been an upsurgence in their popularity in signal processing domains as well. The main attraction is the improved efficiency coupled with decreased power consumption when compared to a GPGPU or a pure software based approach.

To our FPGA we connect 6 ADMP421 omnidirectional MEMS microphones manufactured by Analog Devices (see Figure 5.1). MEMS (MicroElectrical-Mechanical System) technology refers to a class of electronic devices made up of components that are very small ($< 0.1mm$ each). To make devices at this scale not only did there have to be a revolution in the manufacturing process, but also a revolution in the design since the physical constraints on the system are much different at this tiny scale.

A MEMS microphone is a microphone etched into silicon that works very similar to a condenser microphone. Recall that a condenser microphone has a pressure sensitive plate that acts as one side of a capacitor and vibrations from air pressure changes cause the distance between plates and thus the voltage maintained across the plates to change. The MEMS microphone operates in the same way, but it also has an analog-to-digital converter (ADC) etched right next to the condenser microphone. Thus, the signal

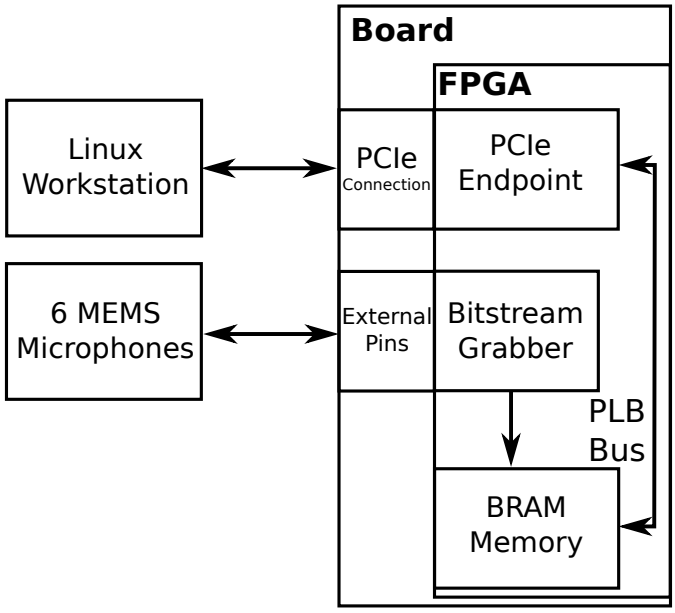


Figure 5.2: Design layout of the FPGA and MEMS microphone audio capture setup.

coming out of the microphone is digital making it much more robust to analog interference. This is why MEMS microphones are recommended for use in devices like cell phones and other small devices where electronic components are packed tightly together causing significant analog interference.

5.2 Hardware and Software Design

The task at hand is to be able to record the digital bitstreams coming from each of the six microphones and store it for future use. The ADMP421 MEMs microphones are attached to the FPGA through the external expansion pins on the board. There are four important connectors per microphone: the data line which carries the digital signal to the FPGA, the clock line which takes the clock from the FPGA to the microphone, and a power and ground lines. In addition, the FPGA is connected to a Linux workstation via a PCIe slot for communicating data in either direction. The primary use of this communication will be for moving the microphone digitized data from the FPGA to the workstation for further processing. A depiction of this setup can be seen in Figure 5.2

The FPGA drives a clock of 2.4 MHz across the pins to each of the 6 MEMS

microphones. When the clock is up, the bitstream coming off each data line is read by the Bitstream Grabber module on the FPGA. This module contains buffers to interleave the bits from each channel and temporarily store them. When these buffers become full, the module then writes them off into a larger block of BRAM memory. This memory is attached to a Processor Local Bus (PLB) bus which connects the BRAM with the PCIe endpoint. The PCIe endpoint can take requests for reading or writing to this block of BRAM from the workstation. This allows the workstation to communicate with the FPGA through the BRAM block and, in particular, fetch the bitstream data across the PCIe connection.

The binary bitstream coming from each of the MEMS microphones is produced by the ADC components of each. The bitstream is a quantized encoding of the full analog signal. The particular encoding used by these microphones is a sigma delta modulation, which requires that the output frequency of the ADC be much higher (2.4 MHz) than the desired final sampling rate of the analog signal (e.g. 16 kHz for our audio processing purposes). If we treat a 0 in the output of the sigma delta modulator as -1 and a 1 as +1, then the average level (taken frame-wise) of this stream represents the analog input signal's level. A high sampling rate is desired by the ADC in order to get good time resolution on the analog signal. In fact, the encoding our MEMS microphones use is a 4th-order sigma-delta modulation, which follows the same principal as described above, but uses feedback loops to better mediate the inherent noise caused due to sampling at such a high rate.

Whenever a recording of the 6 channels of audio is desired, a C program on the workstation is initiated which grabs the bitstream stored in BRAM. The Grabber module writes to BRAM whenever a new 32-bit word of data is ready to be stored. A naive implementation of the workstation program to record the bitstream data would be to simply grab all of BRAM, however, this would only allow for very short recordings since filling up BRAM corresponds to only a fraction of a second of audio recordings. Instead, the workstation continually polls the BRAM at a particular location which denotes that either the 1st half of BRAM is full of fresh data and now it is writing in the 2nd half or vice versa. The workstation polls this flag location in BRAM and grabs the appropriate half of BRAM whenever it changes. This allows for the workstation to continuously

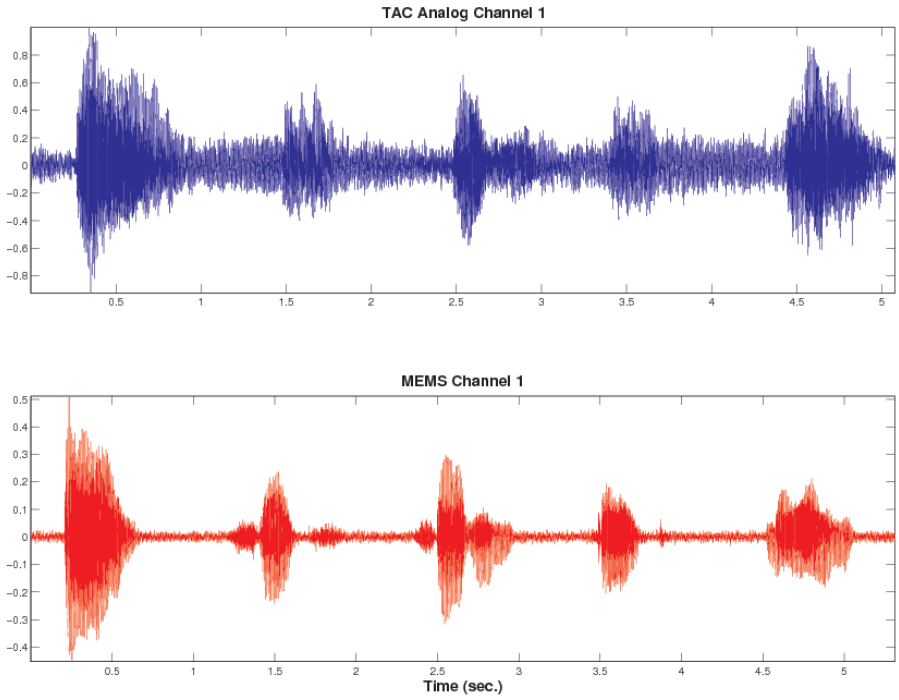


Figure 5.3: Visual comparison of same counting sequence on TAC analog microphone (top) and MEMS microphone (bottom).

grab the bitstream data and allows for recordings that are as long as desired.

It is also worth mentioning that the FPGA board has an Intel AC97 DAC component on it. This allows for 16 bit audio samples to be pushed to it and it will create an analog microphone signal out of the headphone jacks on the FPGA board. By using the FPGAs internal C emulator (microblaze) and implementing the delta-sigma demodulation in the Grabber, this was utilized successfully. The end result is that standard recording programs on the workstation could be used to record a single channel of audio from the MEMS by connecting the headphone jacks between the FPGA and workstation.

5.3 Experiment: Comparison with Analog Microphones

We start with a simple visual comparison of the audio signals between an analog microphone currently being used on TAC and that of the MEMS. A recording of a user

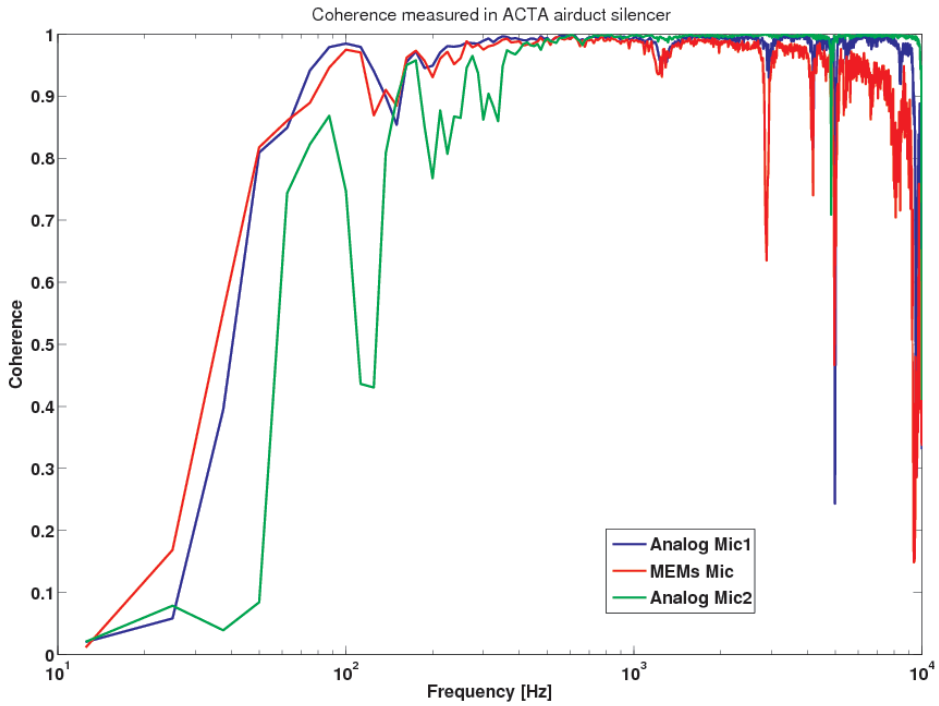


Figure 5.4: Coherence of two different standard analog microphones and our MEMS in a noiseless chamber.

counting was done on each and the results are shown in Figure 5.3. Examining the figure it's visually clear that the background noise level of the microphone is much lower compared to the signal being captured. In fact the SNR in this recording is quite good at 24.7 dB, compared to the results in the beamforming discussion of 6.68 dB.

We also examined the frequency response quality of the MEMS microphone compared to standard analog microphones. With help from Raymond de Callafon's lab in the MAE department at UC San Diego we were able to do this testing in a noiseless chamber for controlling external interferences. The analog microphones we used were not near any electronics equipment that could result in analog interference, so this experiment represents the best case scenario of each type of microphone. A speaker in the chamber repeatedly plays a short segment of white noise, which should have a flat frequency response. The *coherence* for each frequency band is then measured between the known played signal and that on each of the microphone recordings. The coherence

at frequency f is a correlation measure defined by averaging over k measurements as follows,

$$C(f) = \frac{|\sum_i X_i(f)Y_i(f)^*|^2}{\sum_i |X_i(f)X_i(f)^*| \sum_i |Y_i(f)Y_i(f)^*|} \tag{5.1}$$

where X is the reference signal and Y is the measured signal at the microphone. A coherence near 1 at f means that the frequency response at f captures all of the original signal information (up to a scale factor).

The results from this white noise coherence experiment are given in Figure 5.4. The MEMS microphone has a very good response in the frequency range of human speakers (up to ~ 3 kHz), but inferior response at much higher frequencies than its analog counterparts. This may be due to the physical size of the MEMS microphone and the small pressure sensitive hole (see Figure 5.1). Further investigation is needed here to examine why the performance is poor at higher frequencies.

It is also worth noting that the speaker used to produce the white noise was unable to accurately produce the low frequency components of the reference signal. This is why poor performance can be observed in all microphones at these frequencies. Nevertheless, since these microphones are intended for use in small consumer devices, such as cell phones or laptops, there is no need for an accurate response at such low or high frequencies. Human speech is the target for these application and it is well captured by these microphones.

Chapter 6

The Future of TAC

The idea for TAC was a confluence among prototypes being developed simultaneously. Since its initial public unveiling on the 4th floor of the CSE building, many different users have recorded numerous hours worth of videos. However, TAC has remained something of a prototype for sometime and to take it to the next level and attract even more attention is the task moving forward. In this chapter we outline directions for continued research in TAC with special attention paid to how to attract users who want to interact for a longer duration and in greater complexity.

We start by discussing several directions for technical improvements in TAC. First, we discuss how to offload some of the computational load onto the preliminary FPGA work discussed in Chapter 5. Second, we discuss future technical problems that deserve attention in the audio domain with respect to TAC. Lastly, we discuss what we believe the future for user applications with TAC should be including any technical problems that remain to be solved in order to deploy them.

Continuing the MEMS and FPGA Work

The main resource of contention in the current TAC setup is compute cycles. Although the video processing components consume a lot of the CPU cycles, approximately 1 of the 4 CPU cores in the Mac workstation is required for the audio processing. This includes grabbing the audio from the device driver, processing it for PHAT calculation (FFTs), and finally updating the NH-rand particle filter and making a pan and tilt

prediction. During speech periods, all of this computation is performed at a rate of 40 times per second.

By furthering the work with the FPGA much of this computational load can be alleviated from the central Mac workstation. The next obvious step is to implement the PHAT computations on the FPGA itself. The major technical hurdle here is in appropriately implementing the FFTs involved, and in making sure the timing of the entire design is correct. Of course full floating point arithmetic is notoriously difficult to implement efficiently on FPGAs, so a fixed point implementation would have to be employed.

Once PHAT correlation series have been calculated, implementing the NH-rand particle filter should be the next priority. With both of these implemented nearly all of the audio processing could be offloaded from the workstation onto the FPGA. The required information that would need to be passed between the workstation and FPGA would be any directives for pointing the camera and the beamformed DEAS audio for insertion into videos.

By implementing this not only would a core become free on the workstation for other prototyping and testing, but also we would have a much more extensible design for the audio capturing hardware. There would be no restriction on the number of such FPGA units that we could deploy. Ideally, the necessarily communication between the FPGA and the workstation could be done through a wireless network. Then, we could place several audio capture units around the room to get better coverage. Recall the current implementation of TAC is restricted to microphones being connected to the digitizer. By digitizing the signals on the FPGA itself we have not only a mobile digitizer, but also an audio localization unit.

Improved Beamforming and Speech Recognition

Besides the FPGA work, there is also significant room for improvement of other aspects of the audio interaction side of TAC - namely in improved beamforming and speech recognition. Speech recognition in particular is a modality that is severely lacking in TAC. However, both beamforming and speech recognition are tightly intertwined.

There is a significant line of research that attempts to do speech recognition at

a distance through beamforming [JD01, WM09, FAG08, ZLR10]. However, the performance in word error rates of such systems suffer when compared to those with close talking microphones. Nevertheless there is a market for success in automatic speech recognition (ASR) at a distance especially in smart environments and meeting rooms. When multiple speakers are simultaneously talking a microphone array can beamforming to each speaker before performing the ASR whereas a single microphone cannot.

The current implementation of ASR with TAC simply uses the Mac OS X built in speech command recognizer and feeds in the DEAS beamformed audio. The initial attempts were with simple keyword spotting on a small vocabulary consisting of {on,off,left,right,up,down,yes,no,zero,one,two,three}. The performance of the system has an error rate too large to be viably useable. It is worth investing improving the beamforming and possibly a specialized keyword spotting model. Improved control could be utilized such as replacing the on-screen TAC buttons with simple voice commands.

Future User Applications

Despite the discussion above of improving some of the backend technologies of TAC, it is worth emphasizing that the future of TAC should be driven by novel user-application ideas. The audio localization unit had a clear problem to solve: accurately pointing the PTZ camera at human sounds so that the user is contained within the field of view. Simply improving ASR or beamforming, for example, without a particular application goal in mind is a disheartening endeavor. Moreover, it doesn't help to serve TAC's original purpose which is to draw the interest of personnel and other visitors of the UC San Diego CSE building.

An interesting such user-application would be to create a touchless interface that could replace the mouse and keyboard. In fact, this is a current direction of research by the team of graduate students still working on TAC. Tracking points of interest in video accurately and with very fast response are the key to solving this problem. The latter necessity of a fast response time is what makes this problem ideal for an FPGA implementation. By tracking these hand movements, accurate mouse movements and possibly accurate keyboard input using a virtual keyboard becomes possible. The in-

terface is then very similar to a touchscreen device, such as an iPhone, but without the "touch" element. The technical problem of interest is in accurately tracking a series of interest points so that mouse movements feel natural to the user.

An even simpler application that would directly serve the personnel of the building is a touchless music browsing and playing application. Every Friday during the school year the personnel of the building have a social hour directly in front of TAC. Turning TAC into a virtual jukebox during this time is a potential need that is waiting to be filled. The audio localization unit is rendered near useless during this period because of the many different conversations that occur simultaneously (only "near" since this is a very difficult instance of the famous cocktail party problem). However, video processing could be used as an interface for controlling the jukebox, even with the current TAC technology that exists. In fact, even the current on-screen buttons used by TAC could act as an interface to the jukebox. However, ideally something like a touchscreen based phone interface would be nice - swiping gestures to browse through songs, and maybe buttons for pause/play.

These simple directions are nice goals for TAC's future in terms of their ambition and usefulness to the community. By solving these problems successfully, many further doors open up including a holy grail of sorts: more intricate video games. This is a clear direction for making TAC fun and interesting (it is worth noting that a simple game currently exists with TAC by popping bubbles with skin regions). The work with TAC continues on as of the writing of this thesis. Current graduate and undergraduate students will hopefully create new and exciting things for frequenters of TAC's lobby, some of which will hopefully utilize the audio localization technology presented in this dissertation.

Bibliography

[AMGC02] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[BAS95] M.S. Brandstein, J.E. Adcock, and H.F. Silverman. A closed-form method for finding source locations from microphone-array time-decay estimates. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 5:3019–3022, 1995.

[BS05] S.T. Birchfield and A. Subramanya. Microphone array position calibration by basis-point classical multidimensional scaling. *Speech and Audio Processing, IEEE Transactions on*, 13(5):1025–1034, Sept. 2005.

[BVMA09] A. Badali, J.-M. Valin, F. Michaud, and P. Aarabi. Evaluating real-time audio localization algorithms for artificial audition in robotics. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2033 –2038, 10-15 2009.

[CEJ+09] Sunsern Cheamanunkul, Evan Ettinger, Matt Jacobsen, Patrick Lai, and Yoav Freund. Detecting, tracking and interacting with people in a public space. In *ICMI-MLMI '09: Proceedings of the 2009 International Conference on Multimodal Interfaces*, 2009.

[CFH09] K. Chaudhuri, Y. Freund, and D. Hsu. A parameter-free hedging algorithm. In *Advances in Neural Information Processing Systems 22*, pages 297–305. 2009.

[CFH10] Kamalika Chaudhuri, Yoav Freund, and Daniel Hsu. An online learning-based framework for tracking. In *UAI 2010, Proceedings of the 26th Conference in Uncertainty in Artificial Intelligence*, 2010.

[CH94] Y.T. Chan and K.C. Ho. A simple and efficient estimator for hyperbolic location. *Signal Processing, IEEE Transactions on*, 42(8):1905 –1915, aug 1994.

[DF08] Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In *STOC '08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 2008.

[DSY07] Hoang Do, H.F. Silverman, and Ying Yu. A real-time srp-phat source location implementation using stochastic region contraction(src) on a large-aperture microphone array. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 1, pages I–121 –I–124, 15-20 2007.

[EF08] Evan Ettinger and Yoav Freund. Coordinate-free calibration of an acoustically driven camera pointing system. In *ICDSC 2008: Second ACM/IEEE International Conference on Distributed Smart Cameras*, 2008.

[FAG08] Jonathan G. Fiscus, Jerome Ajot, and John S. Garofolo. The rich transcription 2007 meeting recognition evaluation. pages 373–389, 2008.

[FBK+99] W.T. Freeman, P.A. Beardsley, H. Kage, K. Tanaka, C. Kyuman, and C. Weissman. Computer vision for computer interaction. In *ACM SIGGRAPH*, 1999.

[Foy76] W.H. Foy. Position-location solutions by taylor-series estimation. *Aerospace and Electronic Systems, IEEE Transactions on*, AES-12(2):187 –194, march 1976.

[FR95] W.T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *Intl. Workshop on Automatic Face and Gesture Recognition*, 1995.

[Fri87] B. Friedlander. A passive localization algorithm and its accurancy analysis. *Oceanic Engineering, IEEE Journal of*, 12(1):234 – 245, jan 1987.

[FW95] W.T. Freeman and C. D. Weissman. Television control by hand gestures. In *Intl. Workshop on Automatic Face and Gesture Recognition*, 1995.

[GG03] F. Gustafsson and F. Gunnarsson. Positioning using time-difference of arrival measurements. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 6, 6-10 2003.

[GS08] M.D. Gillette and H.F. Silverman. A linear closed-form algorithm for source localization from time-differences of arrival. *Signal Processing Letters, IEEE*, 15, 2008.

[GSS93] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE proceedings. Part F. Radar and signal processing*, 140(2):107–113, 1993.

[HBEM01] Yiteng Huang, J. Benesty, G.W. Elko, and R.M. Mersereati. Real-time passive source localization: a practical linear-correction least-squares approach. *Speech and Audio Processing, IEEE Transactions on*, 9(8):943 –956, nov 2001.

[HLKB05] E. Hörster, R. Lienhart, W. Kellermann, and J.-Y. Bouguet. Calibration of visual sensors and actuators in distributed computing platforms. In *VSSN '05: Proceedings of the third ACM international workshop on Video surveillance & sensor networks*, pages 19–28, New York, NY, USA, 2005. ACM.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[JD01] M. Brandstein J. DiBiase, H. Silverman. *Robust localization in reverberant rooms. In M. Brandstein and D. Ward Microphone Arrays.* Springer-Verlag, 2001.

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

[KC76] C. Knapp and G. Carter. The generalized correlation method for estimation of time delay. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 24(4), aug 1976.

[LJ07] E.A. Lehmann and A.M. Johansson. Particle filter with integrated voice activity detection for acoustic source tracking. *EURASIP J. Appl. Signal Process.*, 2007(1):28–28, 2007.

[LS10] T. Li and W. Ser. Three dimensional acoustic source localization and tracking using statistically weighted hybrid particle filtering algorithm. *Signal Process.*, 90(5):1700–1719, 2010.

[max] Max/msp website. http://www.cycling74.com.

[Mil68] Robert B. Miller. Response time in man-computer conversational transactions. In *AFIPS '68 (Fall, part I): Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 267–277, 1968.

[MLH08] I. McCowan, M. Lincoln, and I. Himawan. Microphone array shape calibration in diffuse noise fields. *Audio, Speech, and Language Processing, IEEE Transactions on [see also Speech and Audio Processing, IEEE Transactions on]*, 16(3):666–670, March 2008.

[MMRC92] J. D. Mackinlay, Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The information visualizer: A 3d user interface for information retrieval. In *Advanced Visual Interfaces, AVI*, pages 173–179, 1992.

[OS94] M. Omologo and P. Svaizer. Acoustic event localization using a crosspower-spectrum phase based technique. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, volume ii, pages II/273 –II/276 vol.2, 19-22 1994.

[OS96] M. Omologo and P. Svaizer. Acoustic source location in noisy and reverberant environment using csp analysis. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 2, pages 921 –924 vol. 2, 7-10 1996.

[PKV08] Pasi Pertilä, Teemu Korhonen, and Ari Visa. Measurement combination for acoustic source localization in a room environment. *EURASIP J. Audio Speech Music Process.*, 2008:1–14, 2008.

[RD04] V.C. Raykar and R. Duraiswami. Automatic position calibration of multiple microphones. *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, 4:iv–69–iv–72 vol.4, 17-21 May 2004.

[RS00] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

[SA87] J. Smith and J. Abel. Closed-form least-squares source location estimation from range-difference measurements. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(12):1661 – 1669, dec 1987.

[Sim] Simulink - simulation and model-based design. `http://www.mathworks.com/products/simulink/`.

[SL06] P. Stoica and Jian Li. Lecture notes - source localization from range-difference measurements. *Signal Processing Magazine, IEEE*, 23(6):63 –66, nov. 2006.

[SMO97] P. Svaizer, M. Matassoni, and M. Omologo. Acoustic source location in a three-dimensional space using crosspower spectrum phase. In *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97) -Volume 1*, page 231, Washington, DC, USA, 1997. IEEE Computer Society.

[SSP05] J.M. Sachar, H.F. Silverman, and W.R. Patterson. Microphone position and gain calibration for a large-aperture microphone array. *Speech and Audio Processing, IEEE Transactions on*, 13(1):42–52, Jan. 2005.

[SYSP05] H.F. Silverman, Ying Yu, J.M. Sachar, and II Patterson, W.R. Performance of real-time source-location estimators for a large-aperture microphone array. *Speech and Audio Processing, IEEE Transactions on*, 13(4):593 – 606, july 2005.

[TPC09] Fotios Talantzis, Aristodemos Pnevmatikakis, and Anthony G. Constantinides. Audio-visual active speaker tracking in cluttered indoors environments. *Trans. Sys. Man Cyber. Part B*, 39(1):7–15, 2009.

[VJ01] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

[VKD09] Nakul Verma, Samory Kpotufe, and Sanjoy Dasgupta. Which spatial partition trees are adaptive to intrinsic dimension? In *UAI 2009, Proceedings of the 25th Conference in Uncertainty in Artificial Intelligence*, 2009.

[WC97] H. Wang and P. Chu. Voice source localization for automatic camera pointing system in videoconferencing. In *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97) -Volume 1*, page 187, Washington, DC, USA, 1997. IEEE Computer Society.

[WLW03] D.B. Ward, E.A. Lehmann, and R.C. Williamson. Particle filtering algorithms for tracking an acoustic source in a reverberant environment. *Speech and Audio Processing, IEEE Transactions on*, 11(6):826 – 836, 2003.

[WM09] Matthias Wölfel and John McDonough. *Distant Speech Recognition*. Wiley, 1. edition, 2009.

[WZsH03] Juyang Weng, Yilu Zhang, and Wey shiuan Hwang. Candid covariance-free incremental principal component analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25:1034–1040, 2003.

[ZLR10] Erich Zwyssig, Mike Lincoln, and Steve Renals. A digital microphone array for distance speech recognition. *In International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 5106–5109, 2010.