Nonlinear Filtering for Narrow-Band Time Delay Estimation

by

Mark M. Tobenkin

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 2009

Author ....
Department of Electrical Engineering and Computer Science
Feb 3, 2009

Certified by ....
Gerald Jay Sussman
Panasonic Professor of Electrical Engineering
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Nonlinear Filtering for Narrow-Band Time Delay Estimation

by

Mark M. Tobenkin

Submitted to the
Department of Electrical Engineering and Computer Science

Feb 3, 2009

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents a method for improving passive acoustic tracking. A large family of acoustic tracking systems combine estimates of the time difference of arrival (TDoA) between pairs of spatially separated sensors – this work improves those estimates by independently tracking each TDoA using a Bayesian filter. This tracking is particularly useful for overcoming spatial aliasing, which results from tracking narrow-band, high frequency sources. I develop a theoretical model for the evolution of each TDoA from a bound placed on the velocity of the target being tracked. This model enables an efficient form of exact marginalization. I then present simulation and experimental results demonstrating improved performance over a simpler nonlinear preprocessor and Kalman filtering, so long as this bound is chosen appropriately.

Thesis Supervisor: Gerald Jay Sussman
Title: Panasonic Professor of Electrical Engineering

Acknowledgments

First and foremost, I'd like to thank J.L., who made a great deal of this work possible.

I want to thank Gerry Sussman for providing me with a wonderful place to work, and people to talk to. This project has gone in a lot of different directions over time, and I greatly appreciate his flexibility, encouragement, and penchant for knowing more about anything I express interest in than I would have ever expected.

Piotr Mitros had an uncanny ability to figure out the causes of my problems, and was an incredibly generous advisor. He is an amazing engineer, and a constant reminder that I should stick to software development.

I am indebted to Dany Qumsiyeh for his patience in listening to my constant stream of consciousness about this project, but also for teaching me most of the approach I now take to problem-solving. He has been a wonderful friend.

I want to thank Michael Mandel for knowing the name and actual theory behind every random problem or idea I happened upon, and for taking the time to help me proof this work.

Mark Feldmeier provided no end of encouragement and circuit design advice for which I'm very grateful.

Alexey Radul kept me on my toes thinking about probability and the many assumptions I made over the course of this work.

Tom Knight provided me with a better understanding of the "bigger picture" that is non-elephant biology.

Finally, I want to thank my parents, and my brother Billy, both for their general support which made any of this possible, and their direct advice and encouragement this last month.

Contents

List of Figures

List of Tables

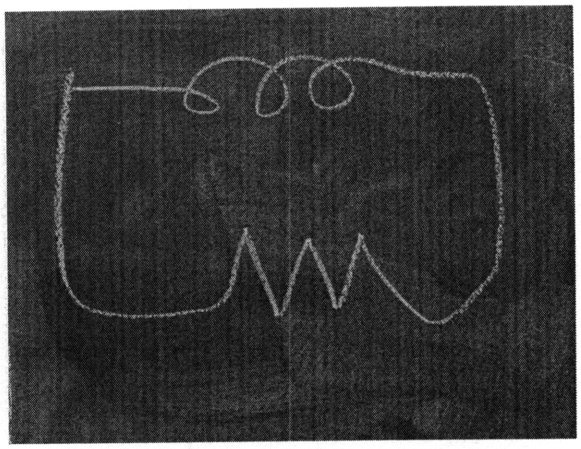
Chapter 1

Introduction

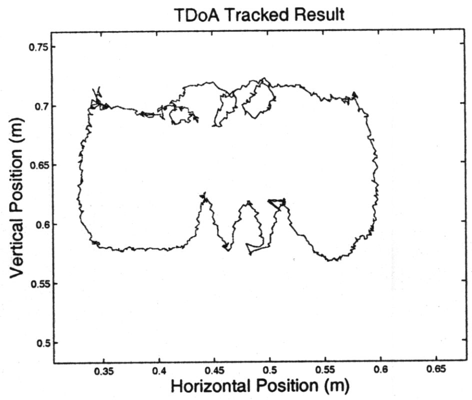
This thesis presents a method for improving passive acoustic tracking. Passive acoustic tracking methods locate a moving target that emits a sound without the benefit of control of or synchronization with this acoustic emission, or the ability to actively probe the target. As a result, passive tracking has a broad set of applications to disparate fields such as seismology, zoology, human computer interfaces and combat.

As sound travels at a finite speed, the acoustic emission of a target arrives at spatially separated sensors at different times — the difference in delay for any two sensors is known as the time difference of arrival (TDoA). A large class of passive tracking systems attempt to estimate the TDoA for multiple pairs of sensors in order to determine the position of the target. Determining the TDoA is particularly difficult for high-frequency, narrow bandwidth signals as multiple TDoAs could be responsible for the same observed data. In this work, I present a method for improving TDoA estimates under these conditions, as a preprocessing stage to determining the target's position.

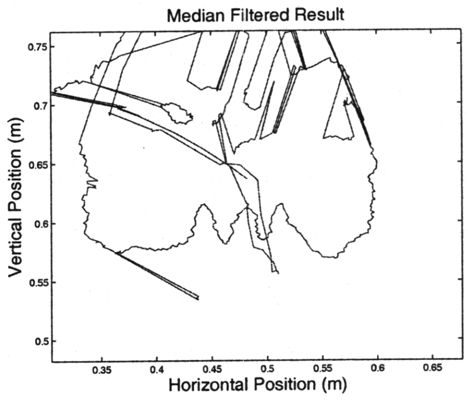
This method involves estimating each sensor pair's TDoA by an independent tracking process. The dynamics of the TDoA evolution are approximated from prior knowledge of the sensor array geometry and a motion model for the target. In particular, I develop a new model for the evolution of the TDoA based on bounding the velocity of the target being tracked. This model allows for efficient exact marginalization.



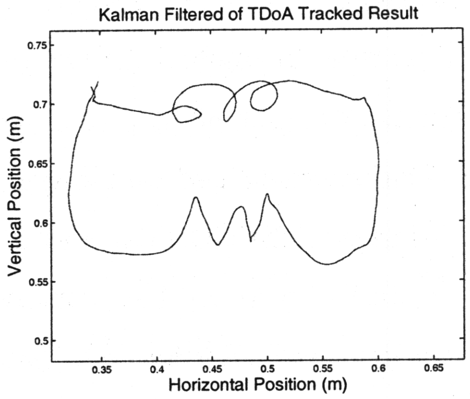
(a) Photograph of Stroke (0.37 × 0.28 m)



(b) Trajectory after TDoA tracking.



(c) Trajectory after median filter preprocessing.



(d) Kalman filter applied to TDoA tracked result.

Figure 1-1: An example of the experimental results from Section 4.3. Figure 1-1(b) presents positions estimated after applying the TDoA Tracking in this work. Figure 1-1(c) demonstrates the failure of a simpler nonlinear preprocessing. Figure 1-1(d) presents the trajectory of Figure 1-1(b) smoothed by a Kalman filter.

I evaluate this TDoA tracking method both via simulation and experiment. The experimental apparatus recovers the trajectories of strokes on a chalk-board by tracking the acoustic emission of the chalk rubbing on the board. Six microphones were used to generate the tracking results in Figure 1-1.

Chapter 2 describes how the phase of incoming signals can be used to estimate the current TDoA. Chapter 3 then develops TDoA tracking from a Markov source model in order to improve these TDoA estimates. Chapter 4 uses simulation and experiment to compare TDoA tracking to simpler nonlinear filters, and applying Kalman filters to the resulting position estimates. Due to the multi-modal uncertainty of TDoA

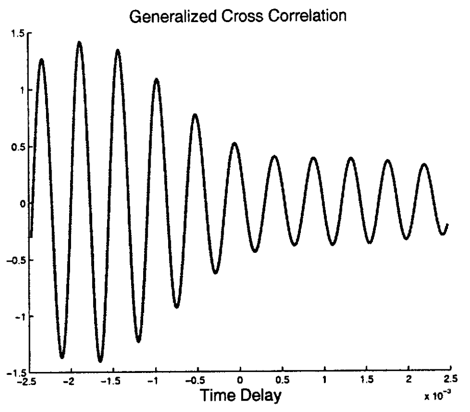


Figure 1-2: An example narrow-band GCC; the signal's narrow bandwidth leads to a plurality of nearby local maxima, or aliases.

estimation from narrow bandwidth sources, TDoA tracking generally outperforms these alternatives for low signal-to-noise ratios. The remainder of this chapter details the scope of the solution provided in this work. Section 1.1 describes the difficulty in estimating TDoA for narrow bandwidth signals and Section 1.2 compares this work to the existing literature.

1.1 Spatial Aliasing

The Generalized Cross Correlation (GCC) of Knapp and Carter [12] is perhaps the most widely used technique for estimating TDoA, and in certain cases provides optimal estimates. The technique amounts to a cross-correlation of the signals arriving at sensors combined with a prefilter. The location of the peak of the correlation is the estimated TDoA. However, determining the location of narrow-band sources can be particularly difficult. Nearby local maxima, known as spatial aliases, can become global maxima in the presence of noise. Figure 1-2 plots an example GCC of a narrow-band signal.

One solution to this problem is to narrow the aperture — that is, the spacing between sensors. The maximum realizable TDoA is proportional to this aperture. If less than a wavelength of the acoustic signal fits between the sensors, then spatial aliases can be avoided. However, for high frequency sources, this can require sensors to

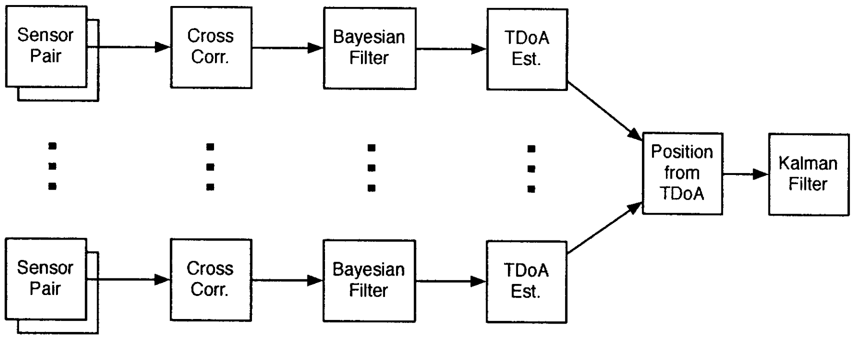


Figure 1-3: A systems level diagram of TDoA tracking.

be so close that very small variance in the TDoA estimation results in wildly different predicted positions. In addition, sensor geometries with randomly distributed sensors cannot guarantee their aperture.

Weiss and Weinstein demonstrate that correlations calculated over longer observations are less affected by these aliases [26]. However, the GCC assumes that the target is not moving. To pretend the target is stationary when it is actually moving, the correlation is commonly evaluated over very short intervals. The faster the target moves, the shorter these intervals must be to make this approximation. This work presents a method for overcoming spatial aliases by combining multiple short observations over a longer interval. Figure 1-3 sketches the systems level diagram of the method proposed in this thesis.

1.2 Prior Work

Early work in acoustic localization focused on determining accurate time delay estimates. For passive detection systems in particular, a great deal of research focused on estimating the time difference of arrival (TDoA) of a signal to multiple pairs of spatially separated sensors. When the signal emitted from the source travels on a straight line path to the sensors (i.e. without reverberation or echoes) the Generalized Cross Correlation [12] has been the estimator of choice for most TDoA based systems. This work focuses on such anechoic environments. [1]

[1]Chen et al. provide a review of techniques for indoor reverberant environments [4].

Bethel and Rahikka make use of a Bayesian filter to improve TDoA estimates in [3]. Similar to the methods in this work, Bethel and Rahikka propagate a discretized posterior distribution for the current time difference of arrival through a first-order Markov model. Their work uses an ad-hoc, but efficient, constant gain structure for propagating posterior distributions forward in time. In this work, we explore the physical motivations for determining these state transitions. Also, we explore Bayesian smoothing, using both past and future observations to improve the current estimate. Finally, we compare methods based not only on the accuracy of time delay estimates, but also on the position estimates generated from those time-delays.

Several previous works improve acoustic localization by other TDoA preprocessing techniques. Allen and King apply an adaptive Kalman-based filter in [1], however their work assumes a broadband source far away from the sensor pair. Tung et al. identify the relationship between continuity of motion and continuity in TDoA [21]. They label the states of a trellis with peaks of the GCC. Transitions between times are weighted with ad-hoc penalties for discontinuity, and based on these costs a smooth path is constructed.

By contrast, this work develops a likelihood for the current TDoA from the GCC – the entire structure of the GCC is used as a powerful indicator of the current TDoA. Also, the transition probabilities for this work are more closely matched to a physical motion model. The approach in this thesis provides reasonable TDoA estimates even if the acoustic source momentarily vanishes.

A variety of techniques exist to transform the approximate TDoAs into a position [8], [18], [20]. Most of these methods require a least one common sensor to every two or three sensor pairs. When this criteria cannot be met, nonlinear optimization is often employed. When dealing with a moving source, position estimates are often smoothed by Kalman Filter or Extended Kalman Filter [23],[11]. At the risk of greatly oversimplifying a large body of research, Figure 1-4 sketches the general system diagram for these approaches. Vermaak and Blake present an interesting alternative approach, combining the determination of position and tracking using Sequential Monte Carlo [22].

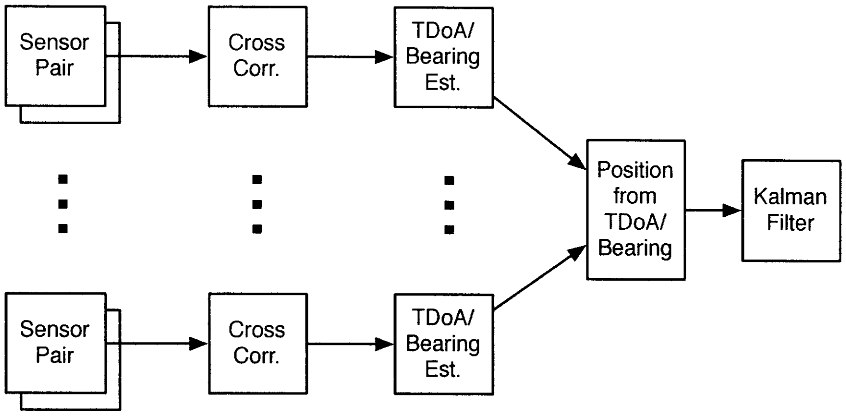


Figure 1-4: A systems level diagram of TDoA localization and Kalman filter tracking.

A different tack for improving estimates for narrow-band sources has been to carefully choose the sensor pairs to minimize the expected error of the TDoA or bearing estimates [27], [11], [23]. Weiss and Weinstein note that below a critical threshold signal-to-noise ratio, the GCC's error explodes. Ash et al. experimentally demonstrate this threshold on a distributed sensor network, and also describe how sensor separation can contribute to lowered signal-to-noise ratios due to radiative attenuation[2]. For a combined network of small-aperture microphone arrays and acoustic intensity sensors, Liu et al. take another interesting approach [14], combining sensor selection, position determination and tracking with a moving locus of sensor fusion in a distributed sensor network. Their tracking problem did not contend with spatial aliases.

Over the last decade attention has shifted to combining the entire observation from all sensors in order to estimate position. One of the predominant techniques is the Steered Response Power algorithm of DiBiase [5]. This technique steers a beam-former over all candidate locations – the greater the energy of the resulting signal, the higher the likelihood that the source is at that position. A variety of structured search algorithms exist to accelerate the process of finding the energy maximizing position [6],[28]. These beam-former techniques have also been combined with Bayesian Filters. For example, Ward et al. apply Sequential Monte Carlo (SMC) to meet real-time performance constraints [25],[24].

In general, methods combining all sensor data should provide better estimates than individually tracking TDoAs. The principal advantage of the proposed method is its parallelism and low dimensionality. Each TDoA pair can be tracked in parallel, based on local information. Furthermore, TDoA tracking is a problem of low enough dimension that exact marginalization is possible.

Chapter 2

Model of Time Difference of Arrival Estimation

2.1 Acoustic Model

This section describes the model for acoustic propagation used by this work. It is similar to that of Knapp and Carter [12], but accounts for a moving source. We assume that there are M sensors with known positions, $\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_M$. The target has a time-varying position $\vec{z}(t)$, and emits a signal $y(t)$ omni-directionally. This signal propagates through space at a fixed speed c. The source signal arrives at each sensor, delayed and attenuated. Because the target moves much slower than the speed of sound, we can approximate the delay using the distance of the source from the sensor when the signal is received:

$$T_i(t) \approx \frac{||\vec{z}(t) - \vec{s}_i||}{c}$$

$T_i(t)$ is called the time of arrival for the i-th sensor. We similarly approximate the attenuation using the distance to the source at the time the signal is received:

$$\alpha_i(t) \approx \frac{1}{||\vec{z}(t) - \vec{s}_i||^2}$$

We model the signal received at the i-th sensor as:

$$x_i(t) = \alpha_i(t)y(t - T_i(t)) + w_i(t)$$

where $w_i(t)$ is some additive noise. Without any prior knowledge of the time domain characteristics of $y(t)$, we cannot generally estimate the time of arrival. We can, however, estimate the difference in delay between a pair of spatially separated sensors. We call this quantity the time difference of arrival (TDoA); between the i-th and ℓ-th sensors, we define the TDoA to be:

$$\tau_{i\ell}(t) \triangleq T_i(t) - T_\ell(t)$$

It is sometimes useful to write the TDoA as a function of the position of the source, rather than time:

$$\mathcal{T}_{i\ell}(\vec{z}) \triangleq \frac{1}{c}\left(||\vec{z} - \vec{s}_i|| - ||\vec{z} - \vec{s}_\ell||\right)$$

So, we have $\tau_{i\ell}(t) = \mathcal{T}_{i\ell}(\vec{z}(t))$. It is important to note that the TDoA is bounded by the aperture, or distance, between the two sensors:

$$\begin{aligned}
\mathcal{T}_{i\ell}(\vec{z}) &= \frac{1}{c}(||\vec{z} - \vec{s}_i|| - ||\vec{z} - \vec{s}_\ell||) \\
&\leq \frac{1}{c}(||\vec{z} - \vec{s}_\ell|| + ||\vec{s}_\ell - \vec{s}_i|| - ||\vec{z} - \vec{s}_\ell||) = \frac{||\vec{s}_\ell - \vec{s}_i||}{c} \\
\mathcal{T}_{i\ell}(\vec{z}) &= \frac{1}{c}(||\vec{z} - \vec{s}_i|| - ||\vec{z} - \vec{s}_\ell||) \\
&\geq \frac{1}{c}(||\vec{z} - \vec{s}_i|| - (||\vec{z} - \vec{s}_i|| + ||\vec{s}_i - \vec{s}_\ell||)) = -\frac{||\vec{s}_i - \vec{s}_\ell||}{c}
\end{aligned}$$

Both the upper and lower bound can be met if the target is collinear with the sensors. This distance is important enough to the problem of estimating the TDoA that we define a short-hand:

$$L_{i\ell} \triangleq \frac{||\vec{s}_i - \vec{s}_\ell||}{c}$$

2.2 Likelihood Model

The goal of TDoA tracking is to determine the time difference of arrival at a set of times $t_0 < t_1 < \ldots < t_n$ using observations of the sensor signals $x_i(t)$ and $x_\ell(t)$. A crucial part of such a probabilistic tracking system is determining the likelihood — the probability of the observed data conditioned on the TDoA.

We determine this likelihood term from a short-term Generalized Cross Correlation (GCC). The Generalized Cross Correlation method assumes a constant TDoA. When estimating $\tau_{i\ell}(t_k)$, we restrict ourselves to using only the observed data over some short interval of time, centered at time t_k. If we choose the duration of this interval, T_W, to be short enough then we can make two simplifying approximations:

$$T_i(t) \approx T_i(t_k) \quad \alpha_i(t) \approx \alpha_i(t_k) \quad \forall\, t \in \left[t_k - \frac{T_\mathrm{W}}{2}, t_k + \frac{T_\mathrm{W}}{2}\right]$$

We define a random variable $\boldsymbol{\tau}_k = \tau_{i\ell}(t_k)$, to be the value of the TDoA at time t_k[1]. At this time, we informally represent the observation at time t_k (i.e. the signals $x_i(t)$ and $x_\ell(t)$ observed over the interval $\left[t_k - \frac{T_\mathrm{W}}{2}, t_k + \frac{T_\mathrm{W}}{2}\right]$) by a random variable $\mathbf{o}_k$. In these terms, the likelihood this section determines can be written $\mathrm{p}\left(\mathbf{o}_k|\boldsymbol{\tau}_k\right)$.

We choose to treat the Generalized Cross Correlation of Knapp and Carter [12] as an unnormalized log-likelihood. This section describes the probabilistic model that this choice implies, and how that differs from the true distribution under the conditions that Knapp and Carter set out. Section 2.2.1 provides a synopsis of the GCC. Section 2.2.2 then discusses the implications of treating the GCC as a log-likelihood, and some of necessary considerations in doing so.

2.2.1 The Generalized Cross Correlation

The GCC principally involves taking the cross-correlation of two sensor signals combined with a prefilter. Knapp and Carter provide a thorough review of the GCC in their seminal work [12]. In that work, they unify a variety of prefilters and demon-

[1]The majority of this chapter is concerned with the TDoA between a single pair of sensors, so we drop the subscript on $\tau_{i\ell}(t_k)$ for notational simplicity.

strate a maximum likelihood (ML) choice under a set of stationarity and Gaussianity assumptions.

To execute the GCC we take the periodogram of each sensor signal, centered at t_k:

$$\hat{X}_i(j\omega) \triangleq \frac{1}{T_\mathrm{W}} \int_{t_k - \frac{T_\mathrm{W}}{2}}^{t_k + \frac{T_\mathrm{W}}{2}} f(t) x_i(t) e^{-j\omega t}\, dt$$

Where $f(t)$ is an appropriately chosen window function[2]. We can now examine the Generalized Cross Correlation between the i-th and ℓ-th sensors in the frequency domain, and the time domain:

$$\begin{aligned}
\hat{G}(j\omega) &\triangleq \psi(\omega)\hat{X}_i(j\omega)\hat{X}_\ell(-j\omega) \\
g(\tau') &\triangleq \int_{-\infty}^{\infty} \hat{G}(j\omega) e^{j\omega\tau'} d\omega \\
&= \int_{-\infty}^{\infty} |\hat{G}(j\omega)| \cos(\omega\tau' + \angle(\hat{G}(j\omega)) d\omega
\end{aligned}$$

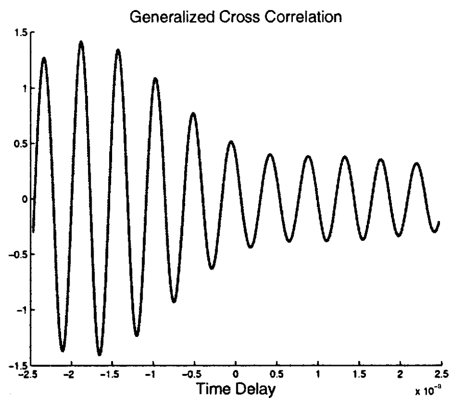
Where $\psi(\omega)$ is the *spectral weighting function*, or prefilter. Roughly speaking, the prefilter decides the degree to which any given frequency is trusted to predict the time difference of arrival. The estimate of the TDoA is the τ at which the GCC achieves a maximum:

$$\hat{\tau}_{i\ell}(t_k) = \underset{\tau' \in [-L_{i\ell}, L_{i\ell}]}{\operatorname{argmax}}\ g(\tau')$$

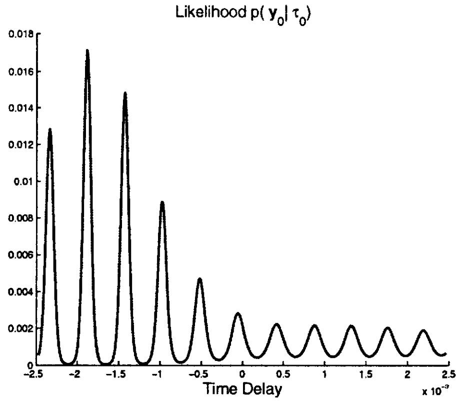
The Hannan-Thomson (HT) and Phase Transform (PHAT) prefilters are particularly noteworthy. The HT choice of prefilter assumes that the sensor signals are jointly stationary, random processes and the cross spectral densities for each of these terms is available. Let S_{x_i,x_ℓ} denote the cross spectral density between signals x_i and x_ℓ and S_{x_i,x_i} denote the power spectral density of a single signal. Then the HT prefilter is given by:

$$\psi_\mathrm{HT}(\omega) = \frac{|S_{x_i,x_\ell}(j\omega)|}{|S_{x_i,x_i}(j\omega)||S_{x_\ell,x_\ell}(j\omega)| - |S_{x_i,x_\ell}(j\omega)|^2}$$

[2]The merits of different windows functions are discussed in [9]. In this work a Hann window is used.



(a) An example Generalized Cross Correlation.



(b) An Example Likelihood.

Figure 2-1: Figure 2-1(a) plots an example Generalized Cross Correlation, evaluated over $[-L_{i\ell}, L_{i\ell}]$. Here a HT filter was used on a narrow-band signal corrupted by small amounts of broadband noise. Figure 2-1(b) plots the same data exponentiated.

Knapp and Carter show the GCC with the HT prefilter to be a Maximum Likelihood estimator for the TDoA under the assumption that the signal and noise processes are zero-mean and Gaussian.

Since complete information about the cross-spectral densities is often unavailable, other ad-hoc techniques are common in practice. The Phase Transform whitens the signal, treating each frequency equally. Technically, the PHAT still assumes access to the cross spectral density:

$$\psi_{\text{PHAT}}(\omega) = \frac{1}{|S_{x_i,x_\ell}(j\omega)|}$$

However, a very common practice is to whiten the periodogram directly:

$$\hat{\psi}_{\text{PHAT}}(\omega) = \frac{1}{|\hat{G}(j\omega)|}$$

2.2.2 The GCC as a Log-Likelihood

For high signal-to-noise ratios, Knapp and Carter demonstrate that the Generalized Cross Correlation is an unnormalized log-likelihood for TDoA [12]. This chapter considers the implications of treating the GCC as such for low signal-to-noise ratios.

To be more concrete, we discuss the choice of modeling:

$$\mathrm{p}\left(\mathbf{o}_k | \boldsymbol{\tau}_k = \tau\right) \propto \exp(Cg(\tau))$$

Where C is an important strictly positive scaling constant, discussed at the end of this section. Figure 2-1 plots an example GCC and the resulting likelihood. In what follows, we consider only a single observation at time t_k, so we use T_i, α_i and $\tau_{i\ell}$ to refer to the time of arrival, attenuation and TDoA at time t_k without ambiguity.

Estimation from Phase without Noise

To build intuition, we first consider determining the TDoA from the phase of a single frequency of the GCC when there is no noise (i.e. taking $w_i = w_\ell = 0$). If we take a long enough window, T_{W}, then we can approximate the sensor periodogram in terms of the periodogram of the source signal, $\hat{Y}$ at time $t_k - T_i$:

$$\hat{X}_i(j\omega) \approx \alpha_i \hat{Y}(j\omega) e^{-j\omega T_i}$$

$$\hat{X}_\ell(j\omega) \approx \alpha_\ell \hat{Y}(j\omega) e^{-j\omega T_\ell}$$

as in [12]. We can then write the difference in phase of the GCC as:

$$\begin{aligned} \hat{G}(j\omega) &\approx \psi(\omega)\alpha_i(\hat{Y}(j\omega)e^{-j\omega T_i})\alpha_\ell(\hat{Y}(-j\omega)e^{j\omega T_\ell}) \\ \angle(\hat{G}(j\omega)) &\approx \angle(\hat{Y}(j\omega)) - \angle(\hat{Y}(j\omega)) - \omega\tau_{i\ell} \qquad \mathrm{mod}\ 2\pi \\ &= -\omega\tau_{i\ell} \qquad \mathrm{mod}\ 2\pi \end{aligned}$$

The phase of the GCC at any given frequency provides information about the TDoA. However, if the frequency is too high the phase does not determine the TDoA uniquely. In particular, any TDoA τ such that:

$$\tau = \tau_{i\ell} + \frac{2\pi m}{\omega} \qquad m = 0, \pm 1, \pm 2, \ldots$$

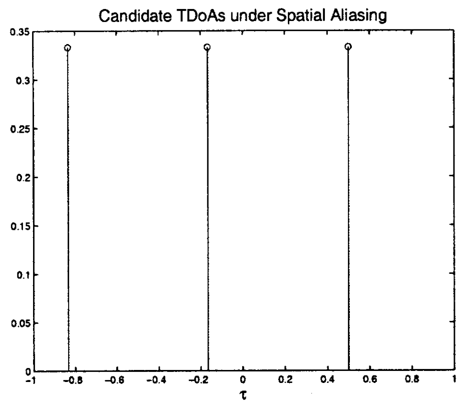


Figure 2-2: The true TDoA and two aliases under spatial aliasing conditions. $L_{i\ell} = 1$, $\omega = 3\pi$ and $\boldsymbol{\tau}_k = \frac{1}{2}$.

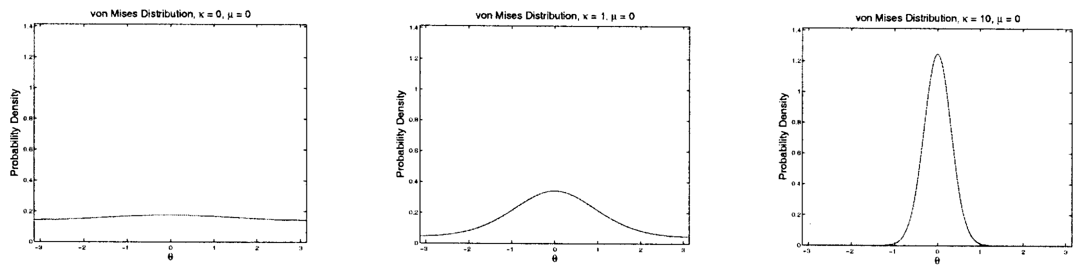
would result in the same phase difference. We need only consider those τ which fit in the bound set by the aperture, $\tau \in [-L_{i\ell}, L_{i\ell}]$. We refer to these alternative TDoAs that result in the same phase difference as spatial aliases. Figure 2-2 plots the true TDoA and two aliases determined in this way. There will be no spatial aliases so long as:

$$\frac{\pi}{\omega} > L_{i\ell}$$

In practice, our periodograms are realized by sampling the sensor signals, which are band-limited, and applying the Discrete Fourier Transform. As a result, we consider only a set of N harmonically spaced frequencies $\{\omega_0, 2\omega_0, \ldots, N\omega_0\}$. We define a set of random variables for the phase of the GCC at these frequencies:

$$\boldsymbol{\theta}_\nu \triangleq \angle\hat{G}(j\nu\omega_0) \quad \boldsymbol{\theta}_{1:N} \triangleq \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$$

For this noiseless case, our $\mathrm{p}(\boldsymbol{\tau}_k \mid \boldsymbol{\theta}_\nu)$ is a series of delta functions, one at the true TDoA, and the others at the spatial aliases.



(a) $\kappa = 0.1$ (b) $\kappa = 1$ (c) $\kappa = 10$

Figure 2-3: The von Mises distribution, an approximate distribution for the phase noise, $\boldsymbol{\phi}_\nu$, for three values of κ and $\mu = 0$.

Estimation from Phase with Noise

We now consider the case where the observed phase difference, $\boldsymbol{\theta}_\nu$, is corrupted by some additive phase noise, $\boldsymbol{\phi}_\nu$. We model these phase noises as independent and von Mises distributed. The choice of the von Mises distribution is purely to be congruent with using the GCC as a log-likelihood, as will be demonstrated shortly. Additional motivation comes from Knapp and Carter [12], who demonstrate that using the HT prefilter this choice tends to the true distribution as the signal-to-noise ratio (SNR) increases, so long as the signal and noise processes are independent, zero-mean, Gaussian and jointly wide-sense stationary.

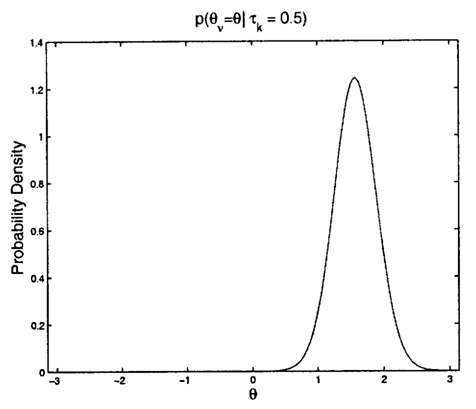
The von Mises distribution is a circular probability distribution characterized by a mean μ and concentration κ:

$$\mathrm{p}\left(\boldsymbol{\phi} = \phi \,|\, \kappa, \mu\right) = \frac{\exp(\kappa \cos(\phi + \mu))}{2\pi I_0(\kappa)} \qquad \phi \in [-\pi, \pi)$$

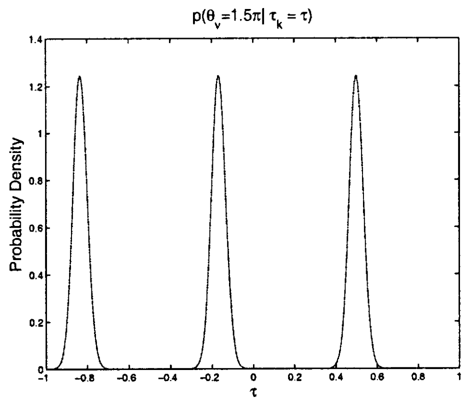
Where I_0 is the modified Bessel function of order zero, and simply a normalizing constant. Figure 2-3 plots the distribution for zero-mean and several values of κ. We take our observed phases to be the phase expected due to the TDoA plus some zero-mean von Mises distributed noise, with concentration κ_ν:

$$\boldsymbol{\theta}_\nu = \nu\omega_0\boldsymbol{\tau}_k + \boldsymbol{\phi}_\nu \qquad \mathrm{mod}\ 2\pi$$

$$\mathrm{p}\left(\boldsymbol{\theta}_\nu = \angle(\hat{G}(j\nu\omega_0)) \,|\, \boldsymbol{\tau}_k = \tau, \kappa_\nu\right) \propto \exp(\kappa_\nu \cos(\nu\omega_0\tau + \angle(\hat{G}(j\nu\omega_0))))$$



(a) $\mathrm{p}\left(\boldsymbol{\theta}_\nu = \theta \,|\, \boldsymbol{\tau}_k = \frac{1}{2}\right)$



(b) $\mathrm{p}\left(\boldsymbol{\theta}_\nu = \frac{3}{2}\pi \,|\, \boldsymbol{\tau}_k = \tau\right)$

Figure 2-4: The likelihood based on a single observed phase, $\mathrm{p}\left(\boldsymbol{\theta}_\nu = \theta \,|\, \boldsymbol{\tau}_k = \tau\right)$, assuming von Mises distributed phase noise. Figure 2-4(a) plots the distribution parameterized as a function of θ. Figure 2-4(b) plots the distribution parameterized as a function of τ with $L_{i\ell} = 1$. In both cases $\nu\omega_0 = 3\pi$ and $\kappa = 10$.

Figure 2-4 plots this likelihood parameterized both by an observed phase and a known TDoA.

Notice that the observed phases are now independent when conditioned on $\boldsymbol{\tau}_k$, leading us to conclude:

$$\begin{aligned}\mathrm{p}\left(\boldsymbol{\theta}_{1:N} \,|\, \boldsymbol{\tau}_k = \tau, \kappa_{1:N}\right) &= \prod_{\nu=1}^{N} \mathrm{p}\left(\boldsymbol{\theta}_\nu \,|\, \boldsymbol{\tau}_k = \tau, \kappa_\nu\right) \\ &\propto \exp\left(\sum_{\nu=1}^{N} \kappa_\nu \cos(\nu\omega_0\tau + \angle(\hat{G}(j\nu\omega_0\tau))\right)\end{aligned}$$

If we choose $\kappa_\nu = C|\hat{G}(j\nu\omega_0)|$, then our final expression is the magnitude-angle form of the Fourier Series of the GCC, exponentiated. As an interesting example, for the approximate PHAT filter, we have $\kappa_\nu = C$.

If we treat this set of measured phases as our observation (i.e. $\mathbf{o}_k = \boldsymbol{\theta}_{1:N}$) we have:

$$\mathrm{p}\left(\mathbf{o}_k \,|\, \boldsymbol{\tau}_k = \tau\right) \propto \exp(Cg(\tau))$$

Where we have omitted mention of our prior knowledge of the concentrations. From this formal equivalence, we arrive at one possible interpretation of treating the GCC as a log-likelihood. We are inferring the TDoA from the phase difference of each frequency of our observation and assuming that each such phase difference is corrupted by independent, zero-mean, von Mises distributed noise.

In [16], Mandel and Ellis use a higher order model based on exponentiating trigonometric polynomials to fit an empirically determined phase noise model based on convolutive noise. This work was later applied to an Expectation Maximization algorithm, which also takes into account the relative attenuation (i.e. ratio of α_i to α_ℓ) between sensors [15].

The Scaling Constant

The Generalized Cross Correlation furnishes a likelihood for the TDoA given the current observation. The TDoA tracking in Chapter 3 will combine these likelihoods over time to determine an improved TDoA estimate. The scaling constant, C in the formula:

$$\mathrm{p}\left(\mathbf{o}_k \,|\, \boldsymbol{\tau}_k = \tau\right) \propto \exp(Cg(\tau)) \qquad C > 0$$

plays an important role in how the observations are combined over time. As C approaches zero, the distribution becomes uniform. As a result, the tracking system will trust prior information. Similarly, as C grows arbitrarily large, the distribution approaches a delta function at those τ that maximize the GCC. In this case, the tracking system will trust the current observation, and ignore prior information.

The shape of the GCC is unaffected by the scaling, so this constant is simply an additional consideration required when exponentiating the GCC. For the maximum likelihood conditions they set out, Knapp and Carter provide a choice for C that allows this ad-hoc distribution to tend to the true distribution for high signal-to-noise ratios [12]. However, this choice is of little practical value, as it requires the cross spectral densities needed for the HT prefilter, which includes knowledge of the attenuation of the source to each sensor. Choosing C to normalize the expected or measured power of g has performed well in practice. However, for non-stationary signal sources, this can lead to undesired effects. For example, consider an erasure, where the source signal $y(t)$ briefly vanishes. One might hope that the likelihood would tend toward a uniform distribution in this case, but such a normalization could prevent this. A more complete answer to this problem could combine signal detection with the tracking developed in the next chapter.

Chapter 3

Time Difference of Arrival Tracking

In general, tracking methods will reject noise and aliases which disagree with their motion model. Ideally, one could examine data from every sensor, and then apply exact marginalization to determine the maximum *a posteriori* position. This strategy is too computationally intensive for real-time performance.

This Chapter develops a nonlinear filter which treats estimating each TDoA as an independent tracking problem. Filtering the evolution of the TDoA is a low-dimensional problem, making exact marginalization possible. The motion model for tracking the target in Cartesian coordinates is used to develop a prior on the dynamics of the TDoA evolution. These dynamics help reject spurious estimates due to noise and spatial aliases. The improved TDoA estimates are then combined to estimate position.

For computational tractability, we simplify the model until a recursive formulation is possible. Section 3.1 describes the derivation of the tracking algorithm from several Markov assumptions. We make use of the Generalized Cross Correlation as a log-likelihood for the TDoA at any given moment, as described in Section 2.2. Next, Section 3.2 describes how priors on the motion of the target can be transformed into priors on the dynamics of the TDoA evolution. These algorithms involve complex integrations, which we approximate using a grid-method: we quantize the possible values of the TDoA. Section 3.3 describes the choice of quantization levels and an efficient implementation of the algorithm.

3.1 Tracking Framework

Our goal in this section is to develop a maximum *a posteriori* estimate for the TDoA at a set of times $t_0 < t_1 < \ldots < t_n$ based on observations made up to the time t_n. Recall from Section 2.2 that we defined a random variable $\boldsymbol{\tau}_k = \tau_{i\ell}(t_k)$, to be the value of the TDoA at time t_k and a variable $\mathbf{o}_k$ to represent the observations centered at that time.

In this chapter, we also make use of the short-hand notation of:

$$\boldsymbol{\tau}_{0:k} \triangleq \{\boldsymbol{\tau}_0, \boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_k\} \quad \text{and} \quad \mathbf{o}_{0:k} \triangleq \{\mathbf{o}_0, \mathbf{o}_1, \ldots, \mathbf{o}_k\}$$

to represent sets of consecutive states and observations.

The tracking problem can be formalized as determining the probability distribution $\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:n}\right)$, and taking the estimated TDoA as the maximum *a posteriori* value:

$$\hat{\tau}_{i\ell}(t_k) = \underset{\tau}{\operatorname{argmax}}\, \mathrm{p}\left(\boldsymbol{\tau}_k = \tau \,|\, \mathbf{o}_{0:n}\right)$$

If $k = n$ this is known as a Bayesian *filtering* problem. If $k < n$ it is known as *smoothing* [7].

To simplify the computation involved, we approximate the sequence of variables $\boldsymbol{\tau}_k$ as a Markov process. Section 3.1.1 derives the estimate in the case of filtering (i.e. $k = n$) for a first-order Markov process. Section 3.1.2 describes the estimator for first-order smoothing.

3.1.1 First-Order Filtering

We first consider the case where $k > 0$. To begin, we expand $\mathrm{p}(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k})$ applying Bayes' Rule:

$$\begin{aligned}\mathrm{p}(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k}) &= \frac{\mathrm{p}(\mathbf{o}_k \,|\, \boldsymbol{\tau}_k, \mathbf{o}_{0:k-1})\,\mathrm{p}(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k-1})}{\mathrm{p}(\mathbf{o}_k \,|\, \mathbf{o}_{0:k-1})} \\ &\propto \mathrm{p}(\mathbf{o}_k \,|\, \boldsymbol{\tau}_k, \mathbf{o}_{0:k-1})\,\mathrm{p}(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k-1})\end{aligned}$$

The denominator is a normalization factor which we disregard. For computational simplicity, we treat $\mathbf{o}_k$ as pair-wise independent of other observations when conditioned on $\boldsymbol{\tau}_k$, which gives us:

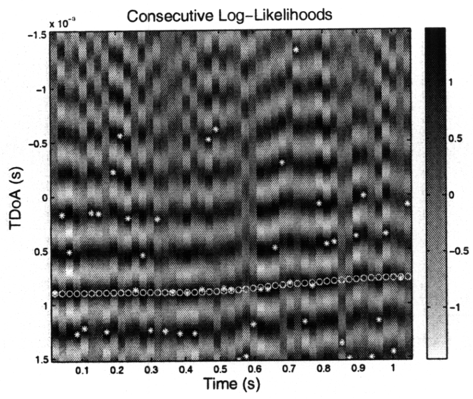
$$\mathrm{p}(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k}) \propto \mathrm{p}(\mathbf{o}_k \,|\, \boldsymbol{\tau}_k)\,\mathrm{p}(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k-1})$$

This independence assumption is contradicted by the fact that if $t_k - t_{k-1} < T_{\mathrm{W}}$ our observations can overlap – however, the system still performs well despite this discrepancy. The first right hand term, $\mathrm{p}(\mathbf{o}_k \,|\, \boldsymbol{\tau}_k)$, is the *likelihood* detailed in Section 2.2. We can expand the second term of the right hand side using the joint distribution with $\boldsymbol{\tau}_{k-1}$:

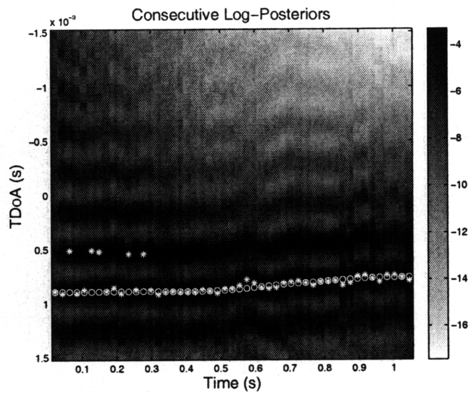
$$\begin{aligned}\mathrm{p}(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k-1}) &= \int \mathrm{p}(\boldsymbol{\tau}_k, \boldsymbol{\tau}_{k-1} = \tau \,|\, \mathbf{o}_{0:k-1})\; d\tau \\ &= \int \mathrm{p}(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1} = \tau, \mathbf{o}_{0:k-1})\,\mathrm{p}(\boldsymbol{\tau}_{k-1} = \tau \,|\, \mathbf{o}_{0:k-1})\; d\tau \qquad (3.1) \\ &= \int \mathrm{p}(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1} = \tau)\,\mathrm{p}(\boldsymbol{\tau}_{k-1} = \tau \,|\, \mathbf{o}_{0:k-1})\; d\tau\end{aligned}$$

The final step of simplification comes from our first-order Markov assumption. This formula gives us a recursive formulation of the *a posteriori* distribution in terms of the likelihood, the previous distribution and $\mathrm{p}(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1})$, our prior on the dynamics of the TDoA evolution:

$$\mathrm{p}(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k}) \propto \mathrm{p}(\mathbf{o}_k \,|\, \boldsymbol{\tau}_k) \int \mathrm{p}(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1} = \tau)\,\mathrm{p}(\boldsymbol{\tau}_{k-1} = \tau \,|\, \mathbf{o}_{0:k-1})\; d\tau$$



(a) Log-Likelihoods



(b) Log-Posteriors

Figure 3-1: An example of First-Order Filtering. Figure 3-1(a) plots consecutive log-likelihoods. Figure 3-1(b) plots consecutive log-posterior distributions. ML and MAP estimates are indicated by stars, and ground truth by circles. An uninformative prior, $\mathrm{p}(\boldsymbol{\tau}_0)$, leads to errors in early estimates.

Section 3.2 will discuss in detail the model for the TDoA dynamics, $\mathrm{p}(\boldsymbol{\tau}_k \mid \boldsymbol{\tau}_{k-1})$.

When $k = 0$ (i.e. when we are examining $\mathrm{p}(\boldsymbol{\tau}_0 \mid \mathbf{o}_0)$) we have the base case of our recursion:

$$\mathrm{p}(\boldsymbol{\tau}_0 \mid \mathbf{o}_0) \propto \mathrm{p}(\mathbf{o}_0 \mid \boldsymbol{\tau}_0)\, \mathrm{p}(\boldsymbol{\tau}_0)$$

Section 3.2 will also discuss the choice of $\mathrm{p}(\boldsymbol{\tau}_0)$. Figure 3-1 plots an example of consecutive likelihoods and posteriors using Bayesian filtering on a simulated TDoA evolution, using the priors suggested in Section 3.2.

3.1.2 First-Order Smoothing

We frame the smoothing problem as finding the TDoA which maximizes the distribution $\mathrm{p}(\boldsymbol{\tau}_k \mid \mathbf{o}_{0:n})$, for each $0 \leq k \leq n$. A more complete answer would find the maximum *a posteriori trajectory* of the TDoA by examining $\mathrm{p}(\boldsymbol{\tau}_{0:n} \mid \mathbf{o}_{0:n})$, but maximizing $\mathrm{p}(\boldsymbol{\tau}_k \mid \mathbf{o}_{0:n})$ for each k independently performs well.

For $k = n$, our solution is given by the filtering problem in the previous section. For $k < n$ we can expand the *a posteriori* distribution applying Bayes' Theorem and our Markov assumptions:

$$\begin{aligned}
\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k}, \mathbf{o}_{k+1:n}\right) &= \frac{\mathrm{p}\left(\mathbf{o}_{0:k} \,|\, \boldsymbol{\tau}_k, \mathbf{o}_{k+1:n}\right) \mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{k+1:n}\right)}{\mathrm{p}\left(\mathbf{o}_{0:k} \,|\, \mathbf{o}_{k+1:n}\right)} \\
&= \frac{\mathrm{p}\left(\mathbf{o}_{0:k} \,|\, \boldsymbol{\tau}_k\right) \mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{k+1:n}\right)}{\mathrm{p}\left(\mathbf{o}_{0:k} \,|\, \mathbf{o}_{k+1:n}\right)} \\
&= \frac{\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k}\right) \mathrm{p}\left(\mathbf{o}_{0:k}\right) \mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{k+1:n}\right)}{\mathrm{p}\left(\boldsymbol{\tau}_k\right) \mathrm{p}\left(\mathbf{o}_{0:k} \,|\, \mathbf{o}_{k+1:n}\right)} \\
&\propto \frac{\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k}\right) \mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{k+1:n}\right)}{\mathrm{p}\left(\boldsymbol{\tau}_k\right)}
\end{aligned}$$

The expression $\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k}\right)$ is also given by the previous section. The same algebra as in (3.1) can also be used to determine $\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{k+1:n}\right)$ recursively:

$$\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{k+1:n}\right) = \int \mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k+1} = \tau\right) \mathrm{p}\left(\boldsymbol{\tau}_{k+1} \,|\, \mathbf{o}_{k+1:n}\right) \, d\tau$$

Where the base-case of this recursion is $\mathrm{p}\left(\boldsymbol{\tau}_n\right)$.

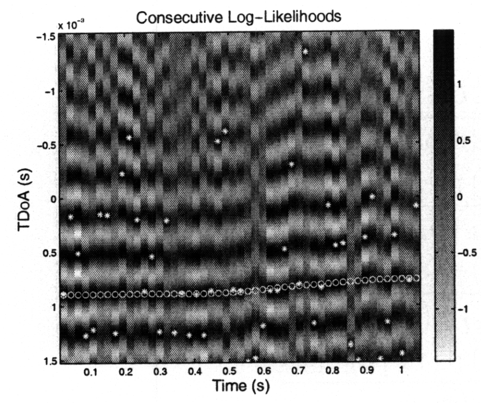
Finally, we must determine $\mathrm{p}\left(\boldsymbol{\tau}_k\right)$. The prior $\mathrm{p}\left(\boldsymbol{\tau}_0\right)$ and the transition probabilities $\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k+1}\right)$ and $\mathrm{p}\left(\boldsymbol{\tau}_{k+1} \,|\, \boldsymbol{\tau}_k\right)$ will be discussed in Section 3.2. Assuming we have these distributions, we can recursively calculate $\mathrm{p}\left(\boldsymbol{\tau}_k\right)$ for $k > 0$:

$$\mathrm{p}\left(\boldsymbol{\tau}_k\right) = \int \mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1} = \tau\right) \mathrm{p}\left(\boldsymbol{\tau}_{k-1} = \tau\right) \, d\tau$$

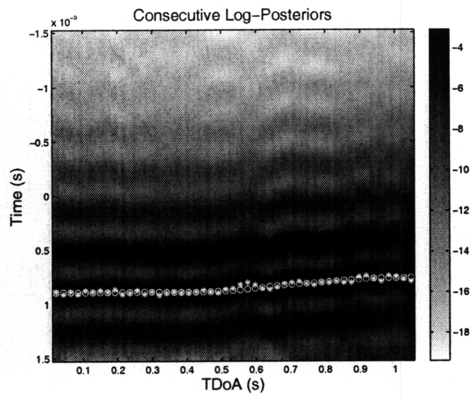
Figure 3-2 plots an example of consecutive likelihoods and posteriors using Bayesian smoothing on a simulated TDoA evolution, using the priors suggested in Section 3.2.

3.1.3 First-Order Partial Smoothing

If the initial prior on TDoA ($\mathrm{p}\left(\boldsymbol{\tau}_0\right)$) is uninformative, early estimates generated by First Order Filtering can be wildly inaccurate. For example, with a uniform prior the first estimate is based purely on the observed data. First-Order Smoothing does not display this problem, at the cost of extra latency and greater resource requirements: the entire history of distributions must be retained. First-Order Partial Smoothing



(a) Log-Likelihoods



(b) Log-Posteriors

Figure 3-2: An example of First-Order Smoothing. Figure 3-2(a) plots consecutive log-likelihoods. Figure 3-2(b) plots the log-posterior distributions. ML and MAP estimates are indicated by stars, and ground truth by circles.

smoothes only the early estimates; as a result it incurs only a fixed amount of extra storage, and latency.

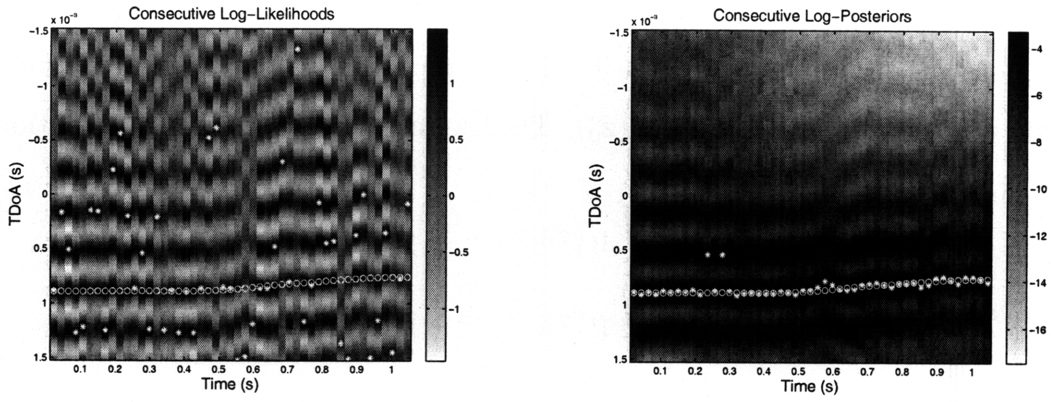
Assume that by the K-th observation, First-Order Filtering has arrived at a meaningful posterior distribution. That is, we expect our estimates:

$$\hat{\tau}(t_k) = \underset{\tau}{\operatorname{argmax}}\, \mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:k}\right) \quad \forall\, k > K$$

to be fairly accurate for all $k > K$. We now smooth our earliest estimates using only the observations up until time K:

$$\hat{\tau}(t_k) = \underset{\tau}{\operatorname{argmax}}\, \mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathbf{o}_{0:K}\right) \quad \forall\, k \leq K$$

to overcome our uninformative prior, $\mathrm{p}(\boldsymbol{\tau}_0)$. Figure 3-3 plots an example of consecutive likelihoods and posteriors using partial smoothing on a simulated TDoA evolution, using the priors suggested in Section 3.2.



(a) Likelihoods (b) Log-Posteriors

Figure 3-3: An example of First-Order Partial Smoothing with $K = 10$ (smoothing of the first 213 ms). Figure 3-3(a) plots consecutive log-likelihoods. Figure 3-3(b) plots the resulting log-posterior distributions. ML and MAP estimates are indicated by stars, and ground truth by circles. Early estimates are improved compared to Figure 3-1.

3.2 Priors

This section discusses the choice of prior distributions for the initial TDoA, $\mathrm{p}(\boldsymbol{\tau}_0)$, and the TDoA dynamics, $\mathrm{p}(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1})$ and $\mathrm{p}(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k+1})$. In particular, it discusses why marginalizing these distributions from a prior distribution over TDoA and position is not necessarily plausible. Instead we provide a simple alternative based on bounding the first difference of the TDoA.

3.2.1 Priors by Marginalization

This section describes, at a high level, a strategy for determining the dynamics of the TDoA from a motion model on the target being tracked. This approach is reasonable when applying TDoA tracking to a sensor array with a known, regular geometry. However, this is not a feasible choice for randomly distributed sensor arrangements. We define a random variable to represent the position of the target at time t_k:

$$\vec{\mathbf{z}}_k \triangleq \vec{z}(t_k)$$

One possible place to start for determining these priors would be a prior on the initial position of the target, or the motion of the target in Cartesian coordinates. Assume we have access to such a $\mathrm{p}\left(\vec{\mathbf{z}}_0\right)$. Then we can write our prior distribution[1]:

$$
\begin{aligned}
\mathrm{p}\left(\boldsymbol{\tau}_0\right) &= \int \mathrm{p}\left(\boldsymbol{\tau}_0, \vec{\mathbf{z}}_0 = \vec{z}\right)\, d\vec{z} \\
&= \int \mathrm{p}\left(\boldsymbol{\tau}_0 \,|\, \vec{\mathbf{z}}_0 = \vec{z}\right) \mathrm{p}\left(\vec{\mathbf{z}}_0\right)\, d\vec{z}
\end{aligned}
$$

The position $\vec{\mathbf{z}}_0$ determines the TDoA exactly; $\mathrm{p}\left(\boldsymbol{\tau}_0 \,|\, \vec{\mathbf{z}}_0 = \vec{z}\right)$ is a delta function:

$$
\mathrm{p}\left(\boldsymbol{\tau}_0 = \tau \,|\, \vec{\mathbf{z}}_0 = \vec{z}\right) = \delta(\tau - \mathcal{T}_{i\ell}(\vec{z}))
$$

Computing this integral analytically would be difficult for any complicated choice of $\mathrm{p}\left(\vec{\mathbf{z}}_0\right)$ (and not necessarily easy even for simple distributions). However, computing this integral numerically, or via a Monte Carlo strategy is plausible, as the TDoA corresponding to any position is easy to compute.

A similar strategy can be taken for determining the dynamics of the TDoA. Assuming access to a joint distribution $\mathrm{p}\left(\vec{\mathbf{z}}_k, \vec{\mathbf{z}}_{k-1}\right)$, we can write the TDoA dynamics as:

$$
\begin{aligned}
\mathrm{p}\left(\boldsymbol{\tau}_k, \boldsymbol{\tau}_{k-1}\right) &= \int\int \mathrm{p}\left(\vec{\mathbf{z}}_k = \vec{z}, \boldsymbol{\tau}_k, \vec{\mathbf{z}}_{k-1} = \vec{z}\,', \boldsymbol{\tau}_{k-1}\right)\, d\vec{z}\, d\vec{z}\,' \\
&= \int\int \mathrm{p}\left(\boldsymbol{\tau}_k, \boldsymbol{\tau}_{k-1} \,|\, \vec{\mathbf{z}}_k = \vec{z}, \vec{\mathbf{z}}_{k-1} = \vec{z}\,'\right) \mathrm{p}\left(\vec{\mathbf{z}}_k = \vec{z}, \vec{\mathbf{z}}_{k-1} = \vec{z}\,'\right)\, d\vec{z}\, d\vec{z}\,'
\end{aligned}
$$

We can then determine the conditional distribution $\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1}\right)$ from this joint distribution. Again, the pair of TDoAs are exactly determined by the positions, so $\mathrm{p}\left(\boldsymbol{\tau}_k, \boldsymbol{\tau}_{k-1} \,|\, \vec{\mathbf{z}}_k, \vec{\mathbf{z}}_{k-1}\right)$ is a bivariate delta function. For similar reasons as before, numerical integration is the simplest strategy.

However, the relationships between the TDoA and position (i.e. $\mathrm{p}\left(\boldsymbol{\tau}_0 \,|\, \vec{\mathbf{z}}_0\right)$ and $\mathrm{p}\left(\boldsymbol{\tau}_k, \boldsymbol{\tau}_{k-1} \,|\, \vec{\mathbf{z}}_k, \vec{\mathbf{z}}_{k-1}\right)$) depend on the position of both sensors. Computing these integrals and determining these relationships in the field for an unknown and possibly

[1]The integrals in this section represent integrals over all the dimensions of the vector valued quantity $\vec{z}$.

uncertain sensor array geometry could result in undesirable startup costs for tracking, and brittleness with respect to errors in absolute position. The next section describes an ad-hoc technique which can be quickly computed based only on local range information.

3.2.2 Bounded Velocity Priors

This section describes a simple alternative based on assuming a bound on the velocity of the object being tracked. Liu et al. use a similar model for developing a Bayesian tracking problem in [14]. Here, however, we transform this bound on the motion model into a bound on the rate-of-change of the TDoA.

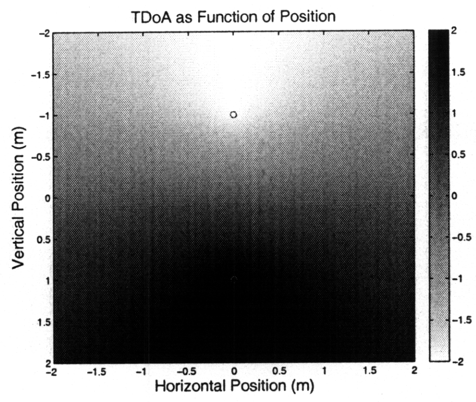
The simplified model for the TDoA dynamics is based on the following observation: if we model the target as having a bounded velocity, then the corresponding rate-of-change of TDoA will be bounded. Let $||\dot{\vec{z}}(t)|| \leq v_{\max}$, for some positive constant $v_{\max}$. Then[2]:

$$\begin{aligned}
\frac{d}{dt}\tau_{i\ell}(t) &= \frac{d}{dt}\left(\mathcal{T}_{i\ell}(\vec{z}(t))\right) \\
&= \left(\nabla \mathcal{T}_{i\ell}(\vec{z}(t))\right)^T \dot{\vec{z}}(t) \\
\left|\frac{d}{dt}\tau_{i\ell}(t)\right| &\leq ||\nabla \mathcal{T}_{i\ell}(\vec{z}(t))||\, v_{\max} \\
&= \frac{1}{c}\left|\left|\frac{\vec{z}(t)^T - \vec{s}_i^T}{||\vec{z}(t) - \vec{s}_i||} - \frac{\vec{z}(t)^T - \vec{s}_\ell^T}{||\vec{z}(t) - \vec{s}_\ell||}\right|\right| v_{\max} \\
&\leq \frac{2}{c} v_{\max}
\end{aligned}$$

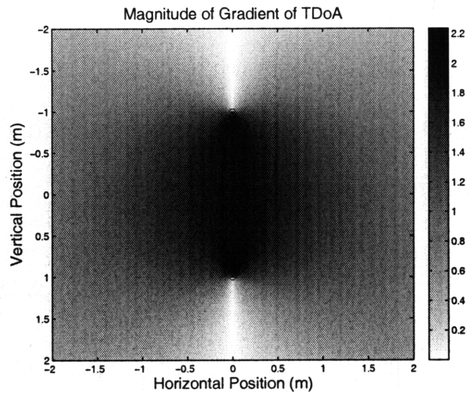
Both the upper and lower bound of $\frac{d}{dt}\tau_{i\ell}(t)$ are achievable if the target moves with speed $v_{\max}$ from one sensor directly toward the other, or vice-versa. Figure 3-4 plots $\mathcal{T}_{i\ell}$ and $||\nabla \mathcal{T}_{i\ell}||$ evaluated over a range of positions.

This bound places a "band-diagonal" constraint on the support of $\mathrm{p}(\boldsymbol{\tau}_k, \boldsymbol{\tau}_{k-1})$.

[2]We use $\cdot^T$ to indicate transpose.



(a) $\mathcal{T}_{i\ell}(\vec{z})$



(b) $\|\nabla \mathcal{T}_{i\ell}(\vec{z})\|$

Figure 3-4: Circles indicate sensors at $(0, 1)$ and $(0, -1)$. Figure 3-4(a) plots TDoA as a function of position with $c = 1\frac{\text{m}}{\text{s}}$. Figure 3-4(b) plots the magnitude of the gradient of TDoA as a function of position.

We know that:

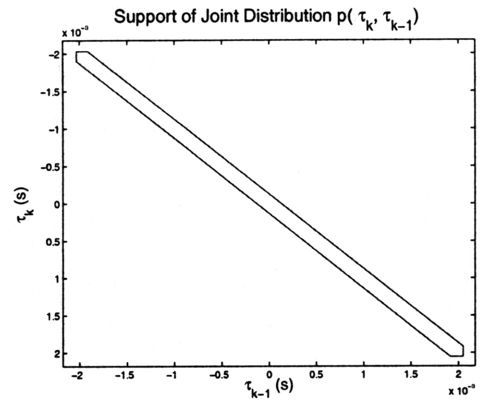
$$\mathrm{p}\left(\boldsymbol{\tau}_k = \tau, \boldsymbol{\tau}_{k-1} = \tau'\right) = 0 \qquad \forall\, \tau, \tau' \text{ s.t. } |\tau - \tau'| > \frac{2v_{\max}}{c}(t_k - t_{k-1})$$

That is, we have bounded the first-difference of $\boldsymbol{\tau}_k$. For many trajectories, this bound is very conservative. If we know that the target being tracked has a limited region of motion which does not include the line segment connecting the two sensors, then a tighter bound can be found by examining $\|\nabla \mathcal{T}_{i\ell}\|$ on this region. Also, it should be noted that it requires a very particular circumstance for this bound to be achieved by two sensor pairs simultaneously. The line segments connecting the two pairs of sensors must be parallel and overlap. When determining position in Chapter 4, this fact will provided added robustness to the position estimates furnished by TDoA tracking.

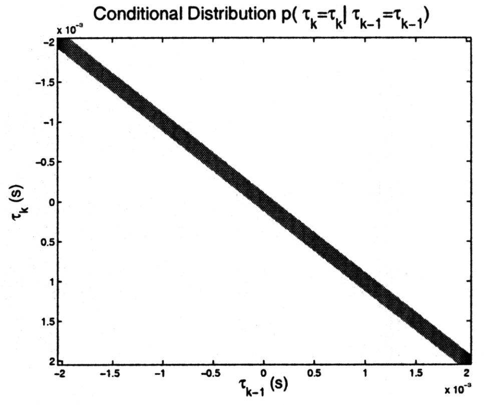
We also have constraints from the aperture of the sensor pair, $L_{i\ell}$:

$$\mathrm{p}\left(\boldsymbol{\tau}_k = \tau, \boldsymbol{\tau}_{k-1}\right) = 0 \qquad \forall\, \tau \text{ s.t. } |\tau| > L_{i\ell}$$

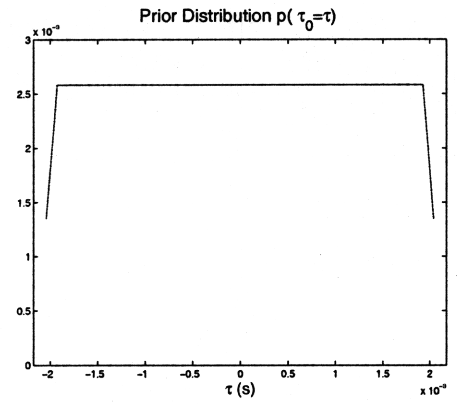
$$\mathrm{p}\left(\boldsymbol{\tau}_k, \boldsymbol{\tau}_{k-1} = \tau'\right) = 0 \qquad \forall\, \tau' \text{ s.t. } |\tau'| > L_{i\ell}$$



(a) Support of $\mathrm{p}\left(\boldsymbol{\tau}_k, \boldsymbol{\tau}_{k-1}\right)$.



(b) Example $\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1}\right)$.



(c) Example $\mathrm{p}\left(\boldsymbol{\tau}_0\right)$.

Figure 3-5: An example set of bounded velocity priors. Figure 3-5(a) plots an outline of the support of the joint distribution. Figure 3-5(b) plots the conditional distribution, assuming the joint distribution is uniform. Figure 3-5(c) plots the prior $\mathrm{p}\left(\boldsymbol{\tau}_0\right)$.

Figure 3-5(a) highlights the region of the joint distribution which can be non-zero. The band-structure arises from the velocity constraint, and the beveled edges from the maximum and minimum possible TDoA.

A simple, and surprisingly effective choice for the joint-distribution is uniform within this band prescribed by the velocity bound and the aperture. For the uniform choice, Figure 3-5(b) plots the resulting conditional distribution, $\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1}\right)$. Given this choice of conditional distribution, the integral for computing $\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \mathrm{o}_{0:k-1}\right)$ has striking similarity to a convolution of $\mathrm{p}\left(\boldsymbol{\tau}_{k-1} \,|\, \mathrm{o}_{0:k-1}\right)$ with a uniformly distributed interval (a "box"), except at the maximum and minimum values of $\boldsymbol{\tau}_k$. We will later exploit this structure to efficiently approximate the integration.

We can also determine $\mathrm{p}(\boldsymbol{\tau}_0)$ from this distribution (an example is plotted in Figure 3-5(c)). Figure 3-6 plots an example of applying this prior on TDoA dynamics to develop a distribution $\mathrm{p}(\boldsymbol{\tau}_k \mid \mathbf{o}_{0:k-1})$. Figure 3-7 then continues the example, plotting a posterior $\mathrm{p}(\boldsymbol{\tau}_k \mid \mathbf{o}_{0:k})$.

Ideally, one could choose the width of the band of the conditional distribution to be smaller than the distance between spatial aliases. While one cannot exert any control over the speed of the target, or the speed of sound, the choice of the sampling interval, $t_k - t_{k-1}$, allows for control over the width of the band. Choosing $(t_k - t_{k-1}) < \frac{c}{2v_{\max} f_{\max}}$, for a given maximum signal frequency $f_{\max}$, will force the band to be narrower than the delay between spatial aliases.

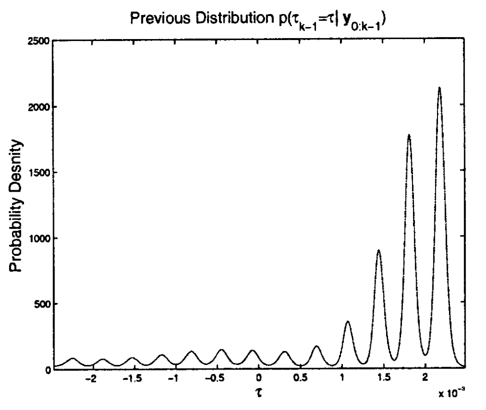
3.3 Implementation

This section details the necessary steps to implement First-Order Filtering using a grid-method for computing integrals. The computation for First-Order Smoothing and Partial Smoothing can be derived similarly. Recall that, in the case of First Order Filtering, our estimates are given as:

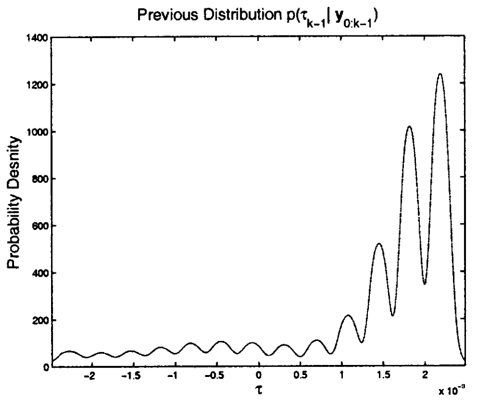
$$\hat{\tau}(t_k) = \underset{\tau}{\operatorname{argmax}}\, \mathrm{p}(\boldsymbol{\tau}_k = \tau \mid \mathbf{o}_{0:k})$$

We break this distribution into a likelihood and recursive prior:

$$\begin{aligned} \mathrm{p}(\boldsymbol{\tau}_k \mid \mathbf{o}_{0:k}) &\propto \mathrm{p}(\mathbf{o}_k \mid \boldsymbol{\tau}_k)\, \mathrm{p}(\boldsymbol{\tau}_k \mid \mathbf{o}_{0:k-1}) \\ &= \mathrm{p}(\mathbf{o}_k \mid \boldsymbol{\tau}_k) \int \mathrm{p}(\boldsymbol{\tau}_k \mid \boldsymbol{\tau}_{k-1} = \tau')\, \mathrm{p}(\boldsymbol{\tau}_{k-1} \mid \mathbf{o}_{0:k-1})\; d\tau' \end{aligned}$$

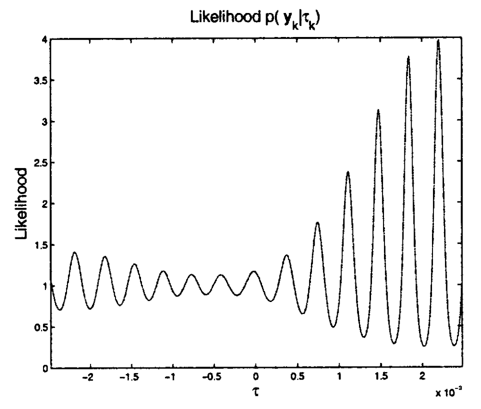


(a) An example $p(\tau_{k-1} \,|\, \mathbf{o}_{0:k-1})$.

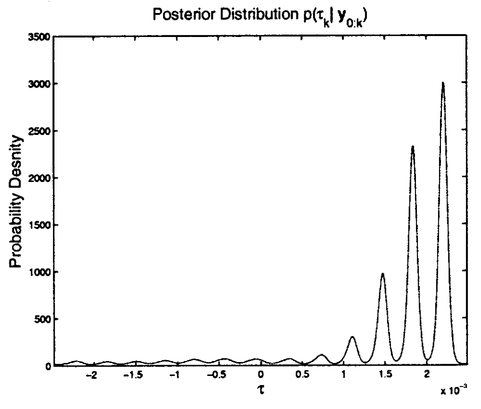


(b) An example $p(\tau_{k} \,|\, \mathbf{o}_{0:k-1})$.

Figure 3-6: Figure 3-6(a) plots an example posterior distribution at step $k - 1$. Figure 3-6(b) plots the prior distribution for the next step, using the bounded-velocity dynamics.



(a) An example $p(\mathbf{o}_k \,|\, \tau_k)$.



(b) An example $p(\tau_k \,|\, \mathbf{o}_{0:k})$.

Figure 3-7: Figure 3-7(a) plots a likelihood based on observation $\mathbf{o}_k$, and Figure 3-7(b) plots the posterior distribution combining the data in Figure 3-6 and this observation. The prior information helps suppress spatial aliases.

To implement this integral, we quantize the possible levels of TDoA to $2Q+1$ separate levels[3]: $\left\{0, \pm\frac{L_{i\ell}}{Q}, \pm 2\frac{L_{i\ell}}{Q}, \ldots, \pm L_{i\ell}\right\}$. Let's define the probability mass functions:

$$\mathrm{p}\left(\tilde{\boldsymbol{\tau}}_k = m \,|\, \mathbf{o}_{0:k}\right) \approx \int_{\left(m-\frac{1}{2}\right)\frac{L_{i\ell}}{Q}}^{\left(m+\frac{1}{2}\right)\frac{L_{i\ell}}{Q}} \mathrm{p}\left(\boldsymbol{\tau}_k = \tau \,|\, \mathbf{o}_{0:k}\right) d\tau \quad m \in \{0, \pm 1, \pm 2, \ldots, \pm Q\}$$

We can represent each of these PMFs as a vector in $\mathbb{R}^{2Q+1}$.

We discretize the integral into a summation:

$$\mathrm{p}\left(\tilde{\boldsymbol{\tau}}_k = m \,|\, \mathbf{o}_{0:k}\right) \propto \mathrm{p}\left(\mathbf{o}_k \,|\, \tilde{\boldsymbol{\tau}}_k = m\right) \sum_{m'=-Q}^{Q} \mathrm{p}\left(\tilde{\boldsymbol{\tau}}_k = m \,|\, \tilde{\boldsymbol{\tau}}_{k-1} = m'\right) \mathrm{p}\left(\tilde{\boldsymbol{\tau}}_{k-1} = m' \,|\, \mathbf{o}_{0:k-1}\right)$$

Where we define the PMF for the dynamics via a similar quantization.

The likelihood vector, $\mathrm{p}\left(\mathbf{o}_k \,|\, \tilde{\boldsymbol{\tau}}_k = m\right)$, is taken as the exponentiation of the samples of a discrete time, prefiltered, linear cross-correlation of the sensor signals. Only the samples corresponding to the range of realizable TDoAs, $[-L_{i\ell}, L_{i\ell}]$, are used. One simple way to choose Q is to set $2Q$ or $2Q+1$ equal to the number of samples within the bounds set by the aperture. If this choice of Q does not provide sufficient accuracy, then the cross-correlation can be re-sampled to accommodate finer quantization levels.

We can also write the computation of the distribution $\mathrm{p}\left(\tilde{\boldsymbol{\tau}}_k \,|\, \mathbf{o}_{0:k-1}\right)$ as a matrix multiplication. Let Π be a matrix in $\mathbb{R}^{2Q+1 \times 2Q+1}$:

$$\Pi_{mm'} = \mathrm{p}\left(\tilde{\boldsymbol{\tau}}_k = m \,|\, \tilde{\boldsymbol{\tau}}_{k-1} = m'\right)$$

then:

$$\mathrm{p}\left(\tilde{\boldsymbol{\tau}}_k \,|\, \mathbf{o}_{0:k-1}\right) = \Pi \cdot \mathrm{p}\left(\tilde{\boldsymbol{\tau}}_{k-1} \,|\, \mathbf{o}_{0:k-1}\right) \tag{3.2}$$

If we take the bounded velocity prior, explained in Section 3.2.2, as an example, we arrive at a band-diagonal matrix.

One of the principal bounds on Q comes from the band-diagonal nature of Π. If

[3]We assume throughout that an odd number of quantization levels is used. Without much modification, $2Q$ levels can be used instead.

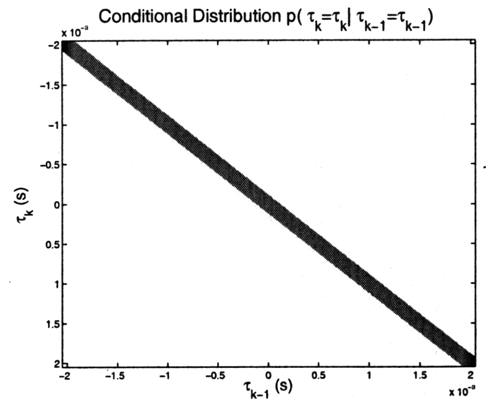


Figure 3-8: The conditional distribution $\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1}\right)$ for bounded velocity dynamics.

Q is chosen to be too small, the band may shrink until Π collapses into an identity matrix. Thus, Q must be chosen sufficiently large to avoid this.

3.3.1 Efficient Computation with Bounded Velocity

Computing this matrix multiplication in (3.2) takes, in general, $O(Q^2)$ steps. With a slight approximation, however, we can compute this update much faster for the bounded velocity dynamics. Figure 3-8 plots again an example of the conditional distribution $\mathrm{p}\left(\boldsymbol{\tau}_k \,|\, \boldsymbol{\tau}_{k-1}\right)$ for the bounded velocity dynamics. The approximation is to compute the PMF of the dynamics $\mathrm{p}\left(\tilde{\boldsymbol{\tau}}_k \,|\, \tilde{\boldsymbol{\tau}}_{k-1}\right)$ by sampling instead of integration, giving us:

$$\Pi_{mm'} = \mathrm{p}\left(\tilde{\boldsymbol{\tau}}_k = m \,|\, \tilde{\boldsymbol{\tau}}_{k-1} = m'\right) \propto \mathrm{p}\left(\boldsymbol{\tau}_k = m\frac{L_{i\ell}}{Q} \,\middle|\, \boldsymbol{\tau}_{k-1} = m'\frac{L_{i\ell}}{Q}\right)$$

Where we normalize each column of Π. We can decompose Π into the product of a diagonal matrix, a convolution matrix and another diagonal matrix. As a result, using the Fast Fourier Transform, we can compute the update in $O(Q\lg(Q))$ steps.

Let W be the maximum width of the matrix band. Note that for a given column of the matrix Π, when computed by sampling, the non-zero elements will all have the same value. Toward the center columns of the matrix, this value is simply $\frac{1}{W}$. For the first and last column, these values are $\frac{1}{1+\frac{W-1}{2}}$. For the m-th column, let this value be c_m.

We can represent Π as another band-diagonal matrix Π', whose every non-zero entry is 1, right multiplied by a diagonal matrix D:

$$D = \begin{bmatrix} c_1 & & & \\ & c_2 & & \\ & & \ddots & \\ & & & c_{2Q+1} \end{bmatrix} .$$

Examining Π', we see that it is the middle rows of the convolution matrix, $H \in \mathbb{R}^{2Q+W \times 2Q+1}$, for convolving a vector in $\mathbb{R}^{2Q+1}$ with a constant vector $h \in \mathbb{R}^W$:

$$h = [1\ 1\ \ldots\ 1]^T$$

We can thus write Π in a block matrix form:

$$\Pi = [\mathbf{0}\ I_{2Q+1}\ \mathbf{0}] \cdot H \cdot D$$

Where each $\mathbf{0}$ consists of $\frac{W-1}{2}$ zero columns[4], and I_{2Q+1} represents the $2Q+1 \times 2Q+1$ identity matrix.

3.3.2 Overall Asymptotic Performance

Let's review the sequence of steps required to compute a single TDoA estimate using First-Order Filtering:

1. Compute the cross-correlation.
2. Re-sample the cross-correlation to fit $2Q$ or $2Q+1$ samples in the bounds set by the aperture.
3. Multiply the previous posterior distribution by the diagonal matrix D.
4. Convolve this matrix product with the vector h.

[4]Note that W will always be odd due to the symmetry of Π.

5. Select the prior vector from the middle elements of this convolution.

6. Compute the posterior vector by point-wise multiplying the likelihood and prior vectors.

7. Determine the index of the maximum value of the posterior.

Step 1 will operate on a number of samples proportional to the observation window duration, T_W, times the sampling rate, f_s. Using the FFT, the linear cross-correlation requires $O((T_W f_s)\log(T_W f_s))$ operations. If the signal is up-sampled using an FIR sinc approximation, step 2 requires only $O(Q)$ time.

Multiplying by the diagonal matrix in step 3 requires $O(Q)$ time. The convolution in step 4 should require only $O(Q\lg(Q))$ time. Selecting the appropriate elements for step 5 requires $O(Q)$ time, and the point-wise multiplication of step 6 and selection of step 7 will likewise require $O(Q)$ time.

The runtime of a single-step of the estimation is thus:

$$O(Q\lg(Q)) + O((T_W f_s)\log(T_W f_s))$$

For each estimate First-Order Smoothing requires a second, similar set of operations, and has the same asymptotic run-time per estimate.

Chapter 4

Simulation and Experiment

This chapter describes the simulations and experiments used to verify the performance of TDoA tracking in comparison to several alternative methods. We contrast the three methods presented in this work — First-Order Filtering, Smoothing and Partial Smoothing — with estimates rendered by the Generalized Cross Correlation, as well as applying a median filter to the GCC's TDoA estimates.

The simulations and experiments are performed for two-dimensional tracking applications. First-Order Smoothing and Partial-Smoothing generally out-perform the other methods in low SNR conditions.

4.1 Determining Position

Both in experiment and simulation, we determine position from TDoA by direct search using the Nelder-Mead simplex method [13]. While a variety of exact and approximate alternative methods exist ([8],[18],[20]), many assume that the TDoA estimates computed involve at least one common sensor to every three estimates. We use the direct search as it is applicable to a more general class of microphone geometries.

Let $\mathcal{P}$ be the set of all pairs of microphones being compared. Recall that $\mathcal{T}_{i\ell}(\vec{z})$ is the function that gives us the TDoA associated with the position $\vec{z}$. The estimated position $\hat{\vec{z}}$ is then approximated by the following minimization:

$$\hat{\vec{z}}(t_k) = \underset{\tilde{\vec{z}}}{\operatorname{argmin}} \sum_{(i,\ell)\in\mathcal{P}} \left(\mathcal{T}_{i\ell}(\tilde{\vec{z}}) - \hat{\tau}_{i\ell}(t_k)\right)^2$$

That is, we find the position which would result in TDoAs closest to the estimated TDoAs in a mean square sense. Sometimes, erroneous TDoAs lead to wildly inaccurate positions. For this reason, we constrain the search to a bounding box.

4.1.1 Kalman Filter

In both experiment and simulation, we make use of a simple inertial Kalman filter to smooth estimated trajectories. In simulation, we present the Kalman filter applied to the estimates generated by the GCC and median filter as alternatives to TDoA tracking. To bias these results in favor of the Kalman filter, we generate the simulated trajectories to match the Kalman filter's model of dynamics, and give the filter access to the true covariance and mean of the measurement noise. For the experimental results, ground truth is not available, so we hand-tune the measurement and process noise covariance to give satisfactory results.

We use the same Kalman filter as [23], which has a simple inertial dynamics and treats accelerations as process noise. The state vector of the Kalman filter is taken to be the two dimensional positions and velocities:

$$q[k] = \begin{bmatrix} \vec{z}(t_k) \\ \dot{\vec{z}}(t_k) \end{bmatrix}$$

$$q[k+1] = Fq[k] + Ga[k] \qquad p[k] = Hq[k] + w[k]$$

$$F = \begin{bmatrix} 1 & 0 & T_a & 0 \\ 0 & 1 & 0 & T_a \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad G = \begin{bmatrix} \frac{1}{2}T_a^2 & 0 \\ 0 & \frac{1}{2}T_a^2 \\ T_a & 0 \\ 0 & T_a \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Where T_a is the time elapsed between estimates. The discrete-time process $a[k]$, whose elements are in $\mathbb{R}^2$, is the Gaussian i.i.d. accelerations. In simulations, the Kalman Filter is given the true disturbance covariance.

4.2 Simulation

This section describes the simulations used to verify the effectiveness of TDoA tracking. We compare a variety of TDoA estimation techniques based on the error of the TDoA estimates and error in the resulting position estimates. The simulation places the microphones and the moving source in a plane. We examine the error over a thousand trials. For each trial, randomized microphone geometries, target trajectories, source signals and noise signals are generated. A wide range of signal-to-noise ratios and both the Hannan-Thomson (HT) and Phase Transform GCC prefilters (see Section 2.2.1) are compared. For low SNR, the First-Order Smoothing technique outperforms the other methods, followed shortly by the First-Order Partial Smoothing.

Though not strictly necessary, the simulation was parameterized as if the discrete time signals were the result of sampling continuous time signals at a rate of 96 kHz. Similarly, the locations are described in terms of meters, and the delays in terms of the speed of sound at sea level; $c \approx 340.29$ was used. The duration of each trial corresponds to tracking a fast moving source for approximately 1.07 seconds using non-overlapping windows of approximately 0.02 seconds.

Section 4.2.1 will discuss the methods being compared by these simulations. Next, Section 4.2.2 will discuss how each random trial is generated. The simulated TDoA tracking makes use of the bounded velocity dynamics presented in Section 3.2, and Sections 4.2.3 and 4.2.4 will present the results of the simulation in terms of error in TDoA estimation and position estimation when the velocity bound is satisfied. Finally, Section 4.2.5 will discuss the performance of TDoA tracking when the velocity bound is violated.

4.2.1 Methods Being Compared

Six sets of TDoA estimates are compared for each trial. First, a quantization of the true TDoA to the sample level is generated as a baseline for comparison. The three methods proposed in this thesis (First-Order Filtering, Smoothing and Partial Smoothing) are computed. Each of these methods makes use of the bounded-velocity prior on the TDoA dynamics, with a maximum velocity $v_{\max} = 1\frac{\text{m}}{\text{s}}$ (see Section 3.2.2). For the TDoA tracking algorithms, we quantize the possible TDoAs to the sampling rate. Thus, the resolution of the TDoA estimates varies with the aperture of the microphone pair.

These methods are compared to the TDoA estimates given by the Generalized Cross Correlation. Also, another dataset is generated by passing the GCC estimates through a nine-tap non-causal median filter. This median filter acts as a nonlinear, constant-time preprocessor to attempt to remove transient errors in the TDoA estimates.

Two measures of estimate quality are used. First, the root mean square error of the TDoA estimates is computed. Next, the TDoA estimates are used to generate position estimates, and the root mean square error of the position estimates is computed.

For comparing accuracy in the final position estimates, we also compare the TDoA tracking techniques to applying a Kalman filter to positions estimated by the GCC and median filtering techniques. Our trial trajectories have been generated to correspond to this model. Section 4.1.1 describes the model.

4.2.2 Trial Generation

The simulation consists of 1,000 trials. Each trial consists of 102,400 samples of a simulated target trajectory and acoustic signal. The acoustic signals are broken into 50, 2048 sample windows, rendering 50 TDoA estimates and a corresponding 50 position estimates.

Each trial consists of an independent trajectory, randomized microphone geometry, and noise and source signals. In what follows, we describe the choices for generating each of these elements of the simulation.

Signal Generation

Recall from Section 2.1 that our signal model is given as:

$$x_i(t) = \alpha_i(t)y(t - T_i(t)) + w_i(t)$$

The source signal $y(t)$ is modeled as shaped white noise. A Gaussian i.i.d. sequence of samples was generated and passed through an 8th-order IIR Butterworth modeling filter. The filter bandwidth is equivalent to 500 Hz with a center frequency of 750 Hz. The noise signals are uncorrelated Gaussian i.i.d. noise processes. A variety of SNRs are compared in the experiment.

The attenuation factor, $\alpha_i(t)$ is modeled as accurately as machine precision allows. The time of arrival, $T_i(t)$, was rounded to the nearest sample. Given the relatively large aperture used in this experiment, this quantization is not expected to have a noticeable effect on the results.

Prefiltering

Two prefilters for the Generalized Cross Correlation are compared. The first is the approximate Phase Transform composed with a "brick-wall" filter from approximately 100-2000 Hz. The second filter is the HT prefilter. For the HT prefilter, we do not assume *a priori* knowledge of the attenuation, and instead normalize the power of the GCC.

Microphone Array Geometry

The simulated sensor array consists of 8 pairs of distinct sensors. Only two distinct pairs are required to generate a unique position estimate. However, additional microphone pairs provide redundancy and robustness to noise. For each trial run a new geometry was generated. The first microphone of each pair was placed with uniform probability in the square with corners $(-1, -1)$ and $(1, 1)$. A random orientation was then chosen, and a second microphone was then placed on a ray emitting from the first in that direction. The aperture of each microphone pair was chosen uniformly on the interval [0.4, 0.8] meters. Figure 4-1 plots an example geometry and trajectory.

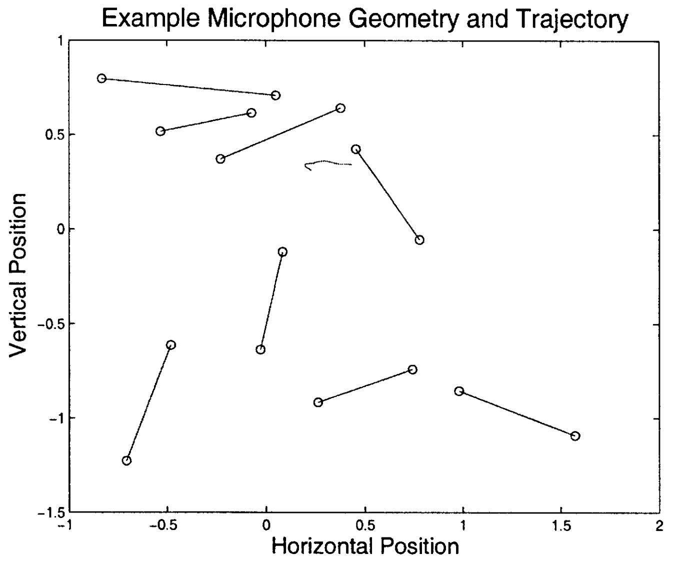
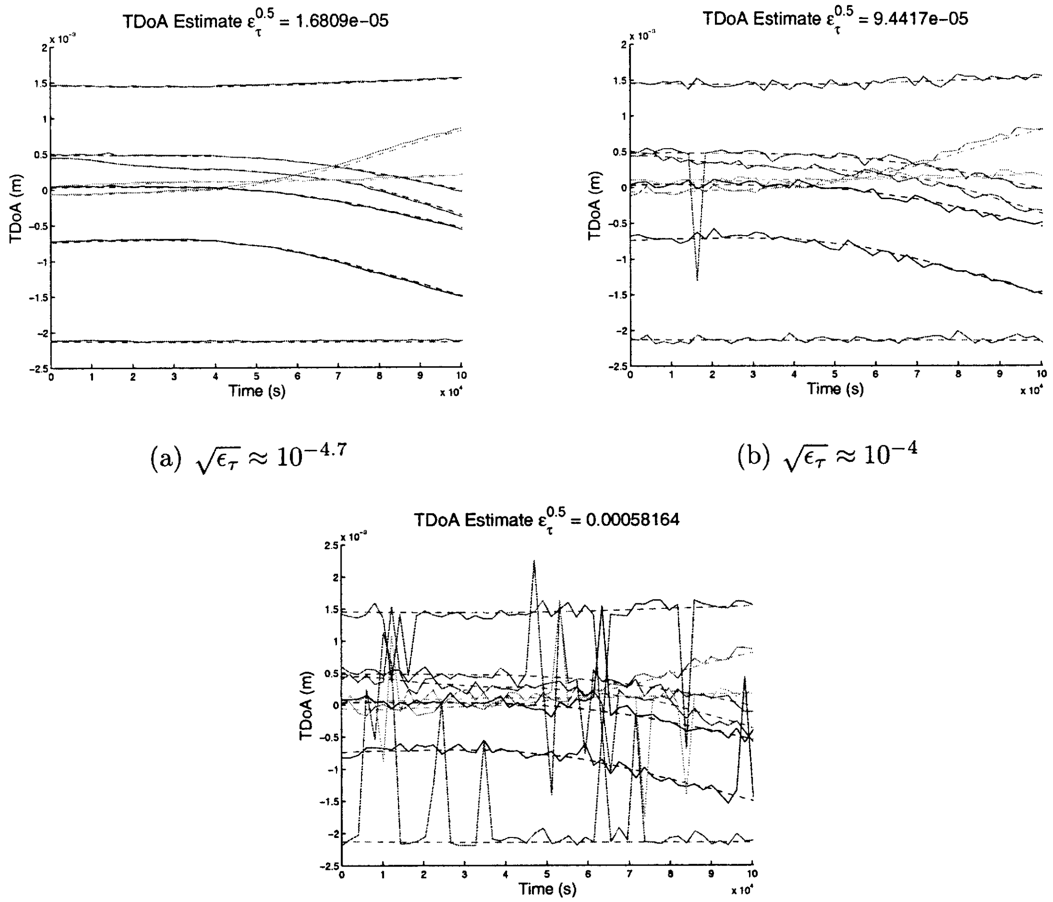


Figure 4-1: An example randomly generated microphone array geometry. Black circles indicate sensors, blue lines indicate the sensors are a pair. The red line indicates an example trajectory.

Trajectory Generation

The sample trajectories, $\vec{z}(t)$, were chosen to allow for easy comparison to a Kalman filter. The initial position and velocities of these trajectories were chosen to be samples from independent Gaussian distributions. For each trajectory a sequence of 50 Gaussian i.i.d. accelerations were generated for the acceleration of both dimensions of the source. A 2048 point zero-order hold was then used to extend the sequence of accelerations over the duration of the simulation. These accelerations were then integrated to give the actual trajectory.



(a) $\sqrt{\epsilon_\tau} \approx 10^{-4.7}$

(b) $\sqrt{\epsilon_\tau} \approx 10^{-4}$

(c) $\sqrt{\epsilon_\tau} \approx 10^{-3.2}$

Figure 4-2: True (dashed) and estimated (solid) TDoAs for various levels of error.

4.2.3 Comparison of TDoA Estimate Error

This section describes the results of these trials in terms of TDoA error. To compare multiple SNRs, we measure the TDoA error by the root mean square error across all estimates. Let $n+1$ be the number of observations in each trial (here 50), and $\mathcal{P}$ be the set of P pairs of microphones being compared. Then the error ϵ_τ is measured as:

$$\epsilon_\tau = \frac{1}{n+1}\sum_{k=0}^{n}\frac{1}{P}\sum_{(i,\ell)\in\mathcal{P}}\left(\tau_{i\ell}(t_k) - \hat{\tau}_{i\ell}(t_k)\right)^2$$

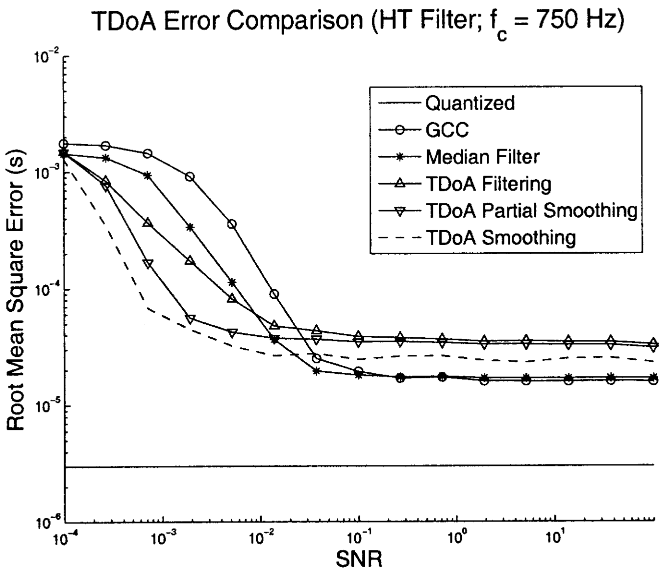
To give some intution for what various levels of error correspond to, Figure 4-2 plots the estimated and true TDoA evolutions for root mean square errors of three different orders of magnitude.

We compare the methods based on the root mean of the error term over all trials. Results using both the Hannan-Thompson (HT) and Phase Transform (PHAT) prefilters were computed over a range of SNRs[1].

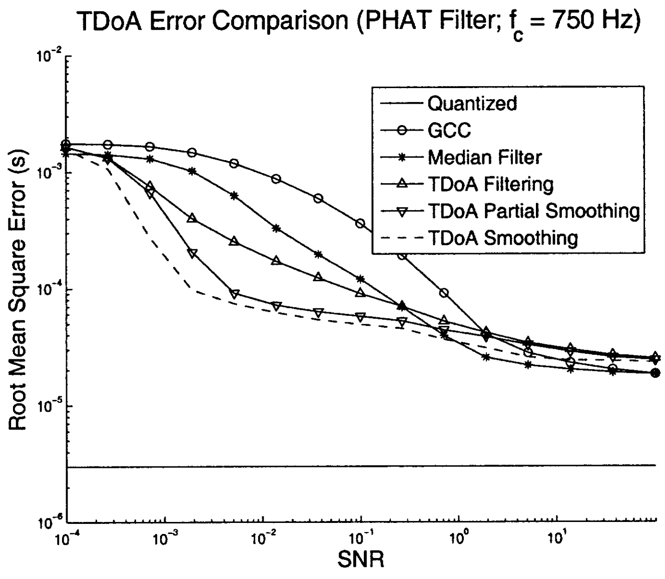
Figure 4-3 compares the mean square errors of the GCC, the GCC placed through a median filter and the Bayesian filtering methods presented in this work. For low SNR, the First-Order Smoothing and Partial Smoothing methods significantly outperform the alternatives. As a baseline, the error due to quantizing the TDoA to the sample level is presented. The HT prefilter displays a distinct threshold signal-to-noise ratio at which the error increases dramatically, as predicted by [26]. For high signal-to-noise ratio, the median filter slightly distorts the already accurate GCC estimates, and barely under-performs the GCC. At the threshold SNR the median filter begins outperforming the GCC, removing occasional errors. Above the threshold SNR both the GCC and median filter out perform the TDoA tracking methods. Section 4.2.4 will compare these same results in terms of position error, and show that this difference in TDoA error does not translate into a significant difference in position error. A very similar structure, without the sharp threshold, is exhibited by the PHAT prefilter simulations. As expected, First Order Partial Smoothing outperforms First-Order Filtering by removing early transients. First-Order Smoothing provides the most marked improvement at low SNRs.

Figure 4-4 presents histograms of the TDoA error for estimates from the GCC, median filter, First-Order Filtering and First-Order Smoothing at the SNR of approximately $10^{-3.1}$ using the HT prefilter. These histograms are taken over all the estimates made over all of the trials for this SNR. The GCC error exhibits multiple modes, as expected due to spatial aliases. Secondary and tertiary aliases give the distribution heavy tails. The median filter successfully rejects of these heavy tails, but actually exacerbates the primary aliases. First-Order Filtering and Smoothing suppress these aliases successfully for this SNR.

[1]The SNR is calculated over the pass-band of the source signal's modeling filter.



(a) HT Filter

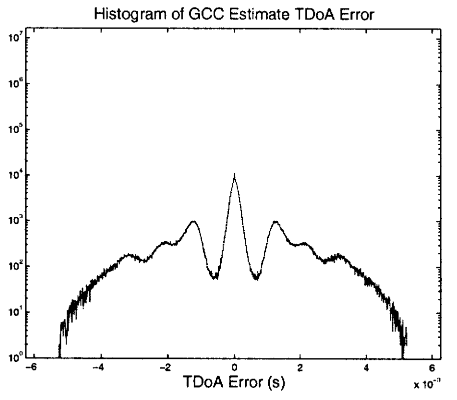


(b) PHAT Filter

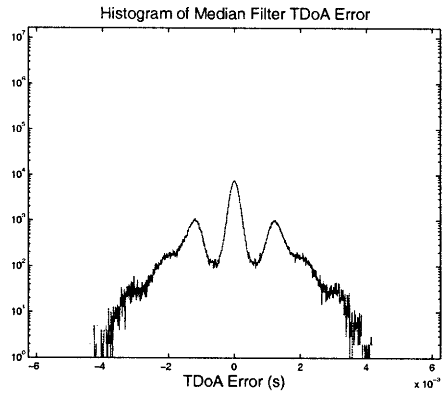
Figure 4-3: Comparison of mean square TDoA estimate error across all trials for various SNR levels, and both the PHAT and HT prefilter.

4.2.4 Comparison of Position Estimate Error

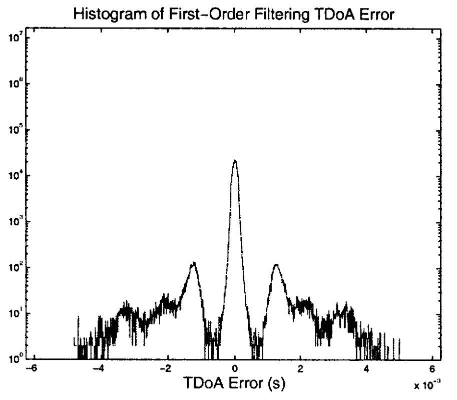
This section describes the results of these trials in terms of error of estimated position. Recall that we determine the estimated positions, $\hat{z}(t_k)$ using the Nelder-Mead optimization as described in Section 4.1. We constrain the search to a similar rectangle



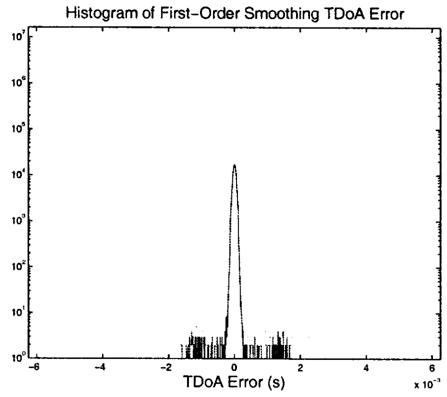
(a) Histogram of GCC Error



(b) Histogram of Median Filter Error

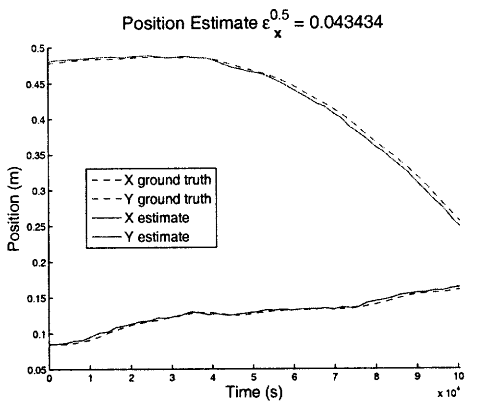


(c) Histogram of First-Order Filtering Error

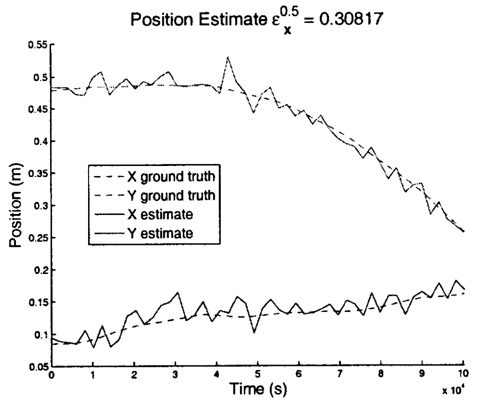


(d) Histogram of First-Order Smoothing Error

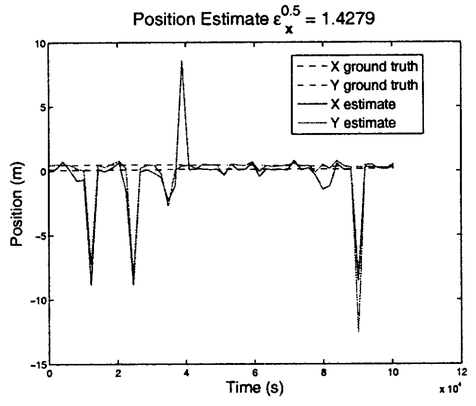
Figure 4-4: Histograms of the TDoA prediction error for the GCC, Median Filter, First-Order Filtering and First-Order Smoothing, at the SNR $\approx 10^{-3.1}$ and using the HT prefilter. The number of estimates with a particular error are plotted on a log scale.



(a) $\sqrt{\epsilon_{\vec{z}}} \approx 10^{-1.4}$



(b) $\sqrt{\epsilon_{\vec{z}}} \approx 10^{-0.5}$



(c) $\sqrt{\epsilon_{\vec{z}}} \approx 10^{0.15}$

Figure 4-5: True and estimated positions for various levels of error.

to the bounding box of the microphone array geometry, with three times the width and height. We summarize the error for multiple SNRs using the mean square error across all estimates. Again, $n+1$ is the number of observations in each trial. Then the error $\epsilon_{\vec{z}}$ is measured as:

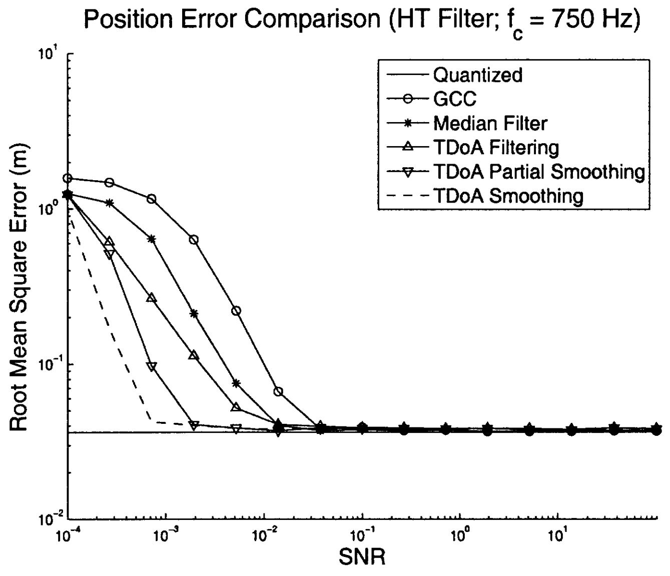
$$\epsilon_{\vec{z}} = \frac{1}{n+1} \sum_{k=0}^{n} ||\vec{z}(t_k) - \hat{\vec{z}}(t_k)||^2$$

We take the root mean of this error term over all the trials for both the HT and PHAT prefilters to compare the different position estimates. Figure 4-5 plots the estimated and true trajectories for errors of three different orders of magnitude.

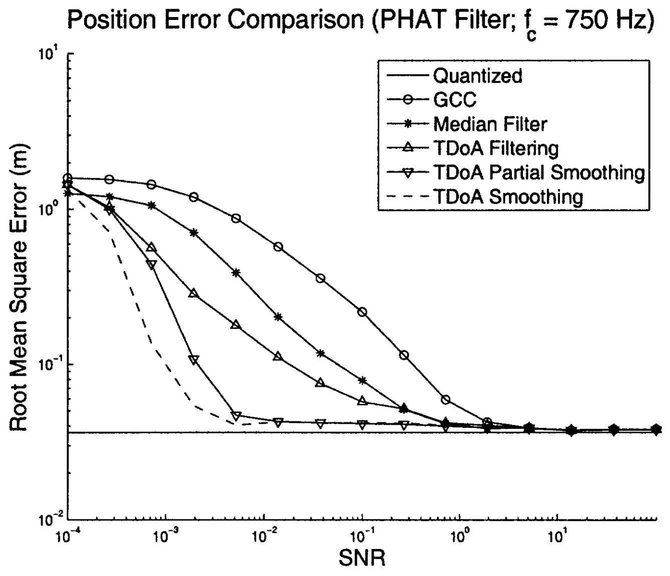
Figure 4-6 plots the mean square errors over all trials for the GCC, median filter and tracking methods. We also take the quantized TDoAs from Section 4.2.3 and use them to predict positions; note that, unlike in Figure 4-3, this line no longer represents an optimum. Similar relationships are observed as when comparing the TDoA errors, though for high SNR the methods do not show dramatic differences in accuracy.

Figure 4-7 plots histograms of the position error over all trials and both dimensions, again for the SNR of approximately $10^{-3.1}$ and the HT prefilter. The median filter's rejection of the heavy tails of TDoA error appears to translate to rejection heavy tails in position error. However, the First-Order Filtering and Smoothing clearly out-perform either, with a sharp peak near zero error.

We now compare the TDoA tracking methods to the GCC and median filter position estimates passed through a Kalman filter. Recall that the target trajectories were generated to match the Kalman filter's motion model, and the true measurement noise covariance and mean are given to these filters. We also plot the result of Kalman filtering the position estimates from the quantization of the true TDoAs. For high SNRs, the noise is in fact unimodal, and this prescient Kalman Filter performs extremely well. First-Order Smoothing outperforms the Kalman filter, near the threshold region of SNRs. The Kalman Filter prevents the error from exploding; the error levels off at $\epsilon_{\bar{z}} \approx 10^0$. It is worth noting that this level of error corresponds to an already wildly inaccurate estimate (see Figure 4-5 on Page 61).

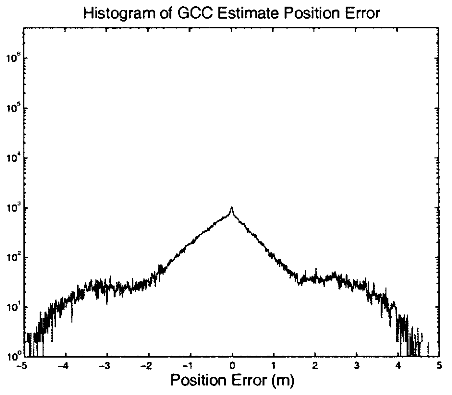


(a) HT Filter

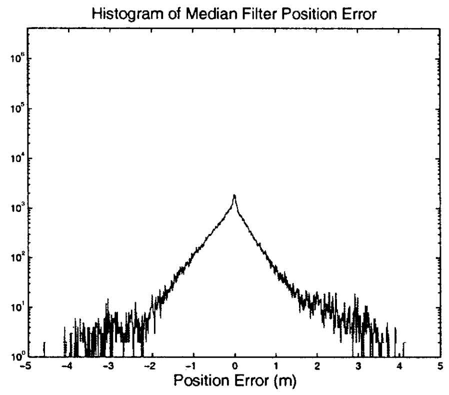


(b) PHAT Filter

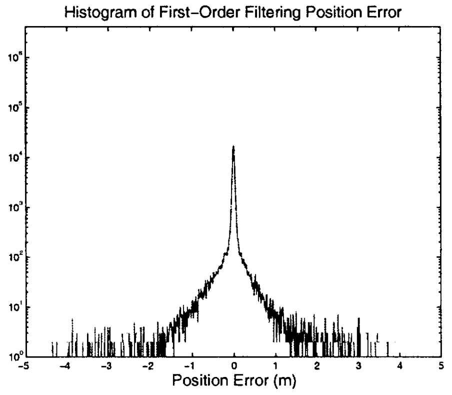
Figure 4-6: Comparison of mean square position error across all trials for various SNR levels, and both the PHAT and HT prefilter.



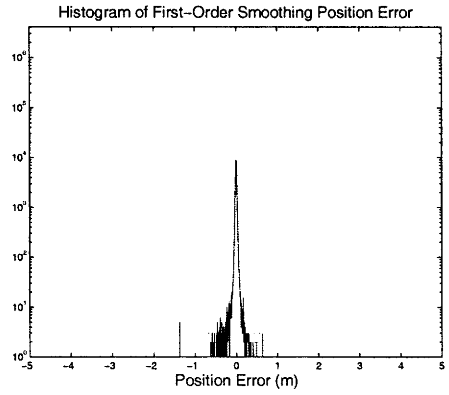
(a) Histogram of GCC Error



(b) Histogram of Median Filter Error

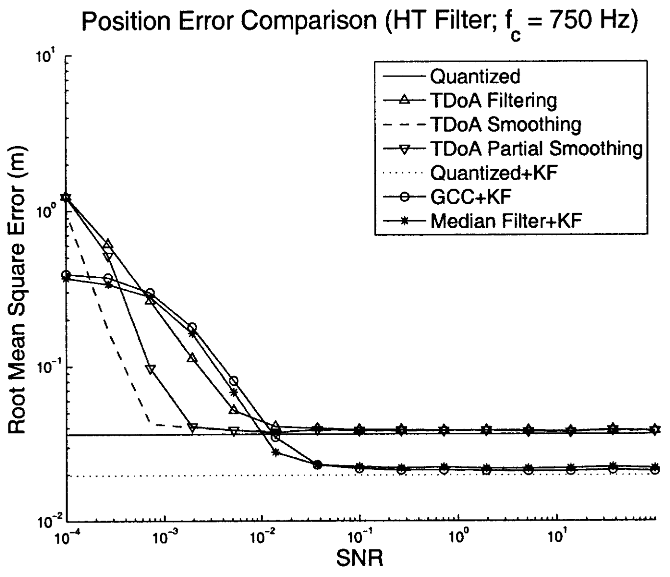


(c) Histogram of First-Order Filtering Error

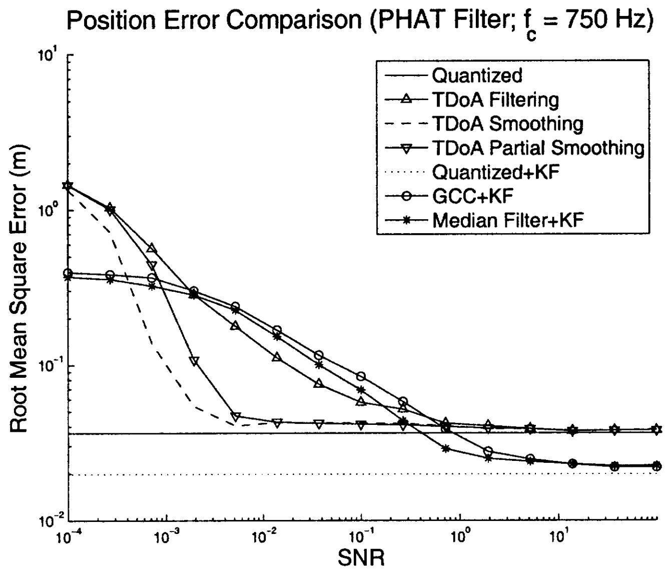


(d) Histogram of First-Order Smoothing Error

Figure 4-7: Histograms of the position estimate error for the GCC, median filter, First-Order Filtering and First-Order Smoothing, for the SNR $\approx 10^{-3.1}$ and the HT prefilter. The number of estimates with a particular error are plotted on a log scale.



(a) HT Filter



(b) PHAT Filter

Figure 4-8: The positions estimated by the TDoA tracking methods compared to the Kalman filter applied to GCC and median filter estimates. The Kalman filter is given the true error covariance and mean, accounting for the large shift down in error.

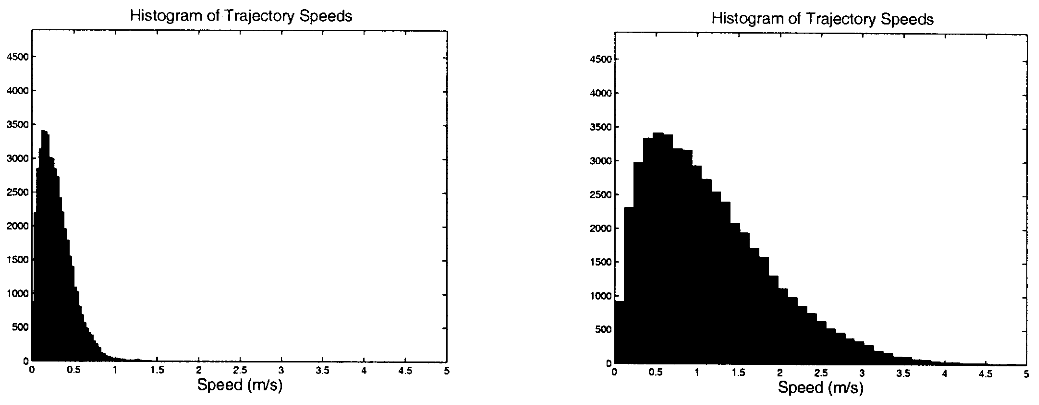


Figure 4-9: Histogram of the speeds attained by the trajectories of the two simulations. The histograms plot the number of ground truth samples which achieve any given speed. Figure 4-9(a) presents the speeds of the simulation results presented earlier in this chapter. A second run generated the speeds plotted in Figure 4-9(b); in this second set of simulations the assumed bound of $1\frac{\text{m}}{\text{s}}$ is often violated.

4.2.5 Velocity Bound Violation

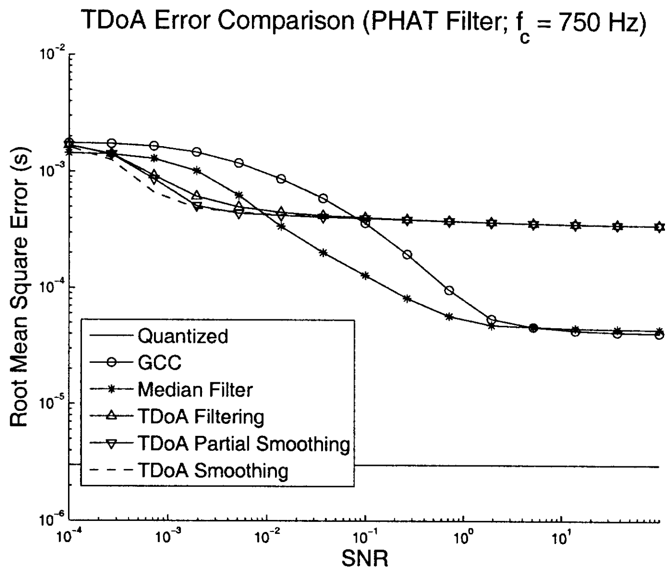
The previous results were obtained using the bounded velocity dynamics, assuming $v_{\max} = 1\frac{\text{m}}{\text{s}}$. Figure 4-9(a) plots a histogram of the velocities actually attained during the thousand trials; for this data-set the velocity bound was very rarely violated. To explore what happens when that bound is violated, a second simulation was run multiplying the acceleration process noise, $a[k]$, by four. Figure 4-9(b) plots the histogram of speeds across the trials of this second experiment. The accuracy of the TDoA tracking estimates were dramatically affected by these occasional violations of the bounds. Figure 4-10(a) plots the error over various SNR to be compared with Figure 4-3 on Page 59.

Figure 4-10(b) plots the position error results, to be compared with Figure 4-6 on Page 63. Surprisingly, the accuracy of position estimates was affected much less by the velocity bounds being violated. For example, at the SNR of 10, the TDoA error is worse by an order of magnitude, whereas the position error is worse by a fraction of that. To explore this relationship at the SNR of $10^{0.29}$, we plot the TDoA error histograms for the GCC and First-Order Smoothing in Figure 4-11. When the velocity bound is violated in this way, First-Order smoothing introduces multi-modal error in the TDoA estimates. This leads to heavier tails in the position estimate errors.

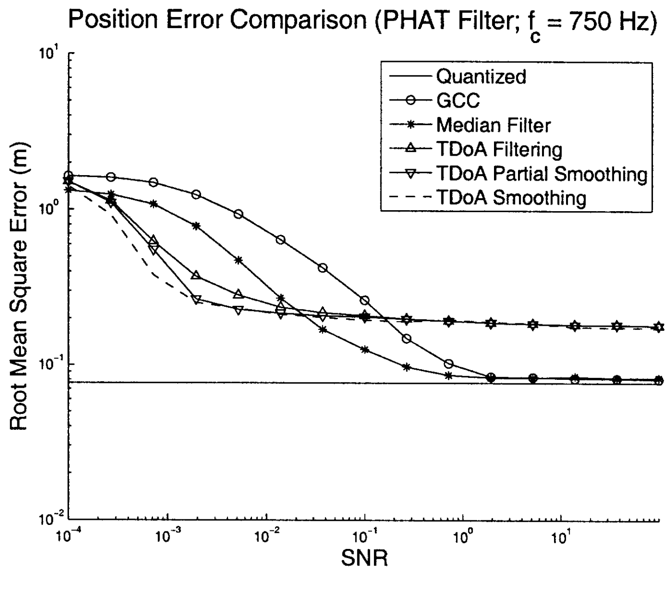
4.2.6 Discussion

When the bounded velocity model closely matches the motion of the target, First-Order Smoothing and First-Order Partial Smoothing outperform the GCC and the median preprocessor for low SNR both in the measures of TDoA and position error. They also outperform First-Order Filtering, but Smoothing provides only a marginal gain over Partial Smoothing. This indicates that most of the error in First-Order Filtering comes from an uninformative prior, which Partial Smoothing corrects (see Section 3.1.3). Plots of the position error shed a particularly favorable light on these two tracking techniques. The Kalman Filter out performs the TDoA tracking techniques, but only for high SNR. Despite having knowledge of the mean of the error, the Kalman filter still underperforms TDoA tracking for low SNR.

The quality of the TDoA estimates rendered by TDoA tracking with bounded velocity dynamics degrades seriously as the bound is violated. However, the position estimates seem to degrade to a lesser degree. We conjecture that, as mentioned in Section 3.2.2, it is uncommon for the violation of the bound to effect more than one microphone pair at a time. As a result, when a single estimate fails, the redunant information available due to the presence of 8 pairs of microphones helps suppress errors.

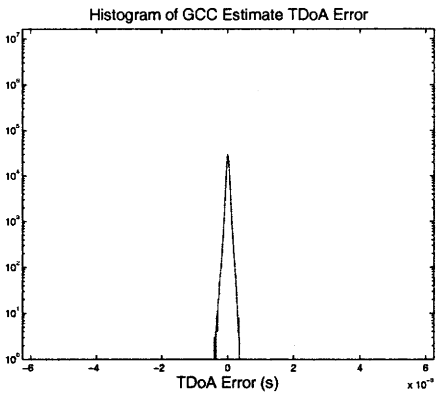


(a) TDoA Error

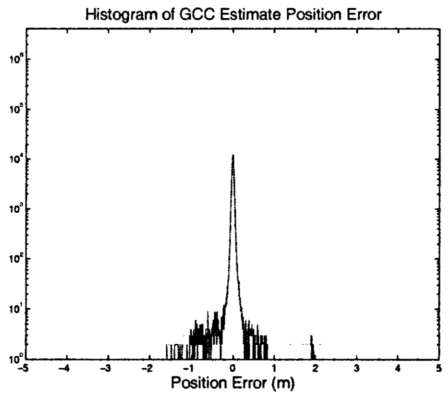


(b) Position Error

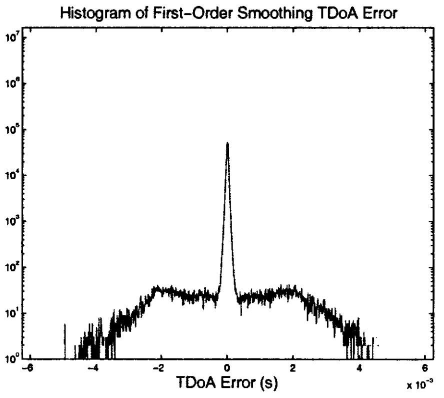
Figure 4-10: Comparison of mean square TDoA error and position error using the PHAT filter across all trials when the velocity bound being violated. The dashed line indicates the SNR used for the histograms in Figure 4-11.



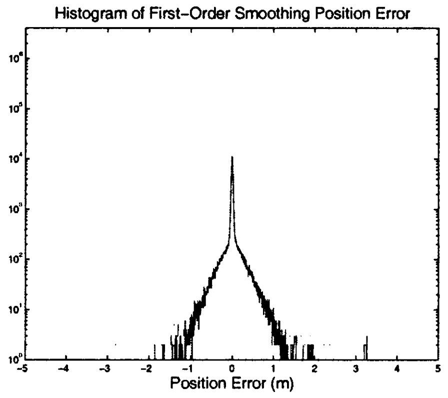
(a) Histogram of GCC TDoA Error



(b) Histogram of GCC Position Error



(c) Histogram of First-Order Smoothing TDoA Error



(d) Histogram of First-Order Smoothing Position Error

Figure 4-11: Histograms of the TDoA and position prediction error for the GCC and First-Order Smoothing with the velocity bound being violated (SNR $= 10^{0.29}$, PHAT prefilter). TDoA tracking introduces aliases into the TDoA errors and heavier tails in position errors. The number of estimates with a particular error are plotted on a log scale.

Table 4.1: Experimental Microphone Geometry

Microphone	X	Y
1	0	0.76
2	0	0.38
3	0.32	0.03
4	0.6	0.03
5	1.02	0.38
6	1.02	0.76

Table 4.2: Experimental Microphone Aperture

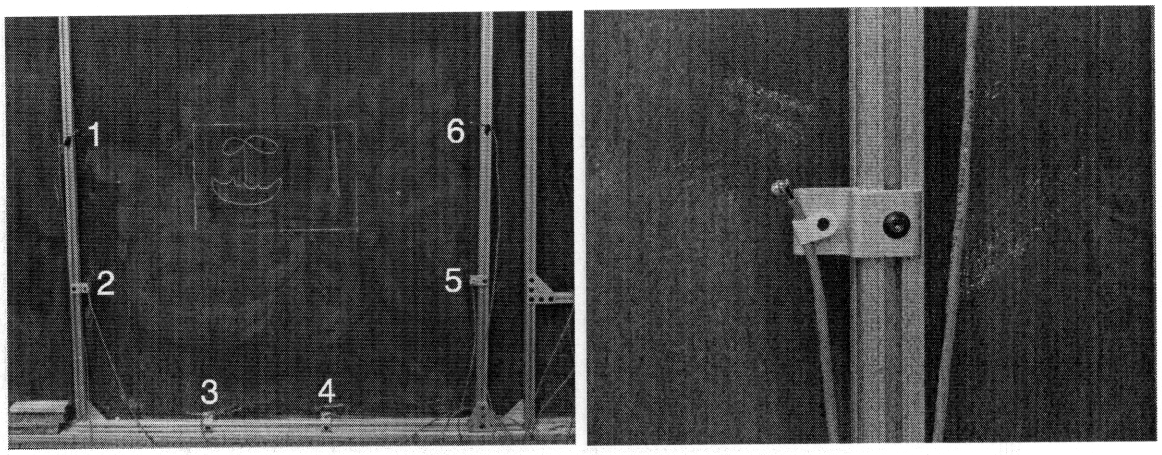
Pair	(3,4)	(1,2)	(5,6)	(2,3)	(4,5)
Aperture (meters)	0.279	0.381	0.381	0.478	0.55
Quantization Levels	158	215	215	269	311

4.3 Experiment

This section describes a series of experiments which demonstrate a real-world application of TDoA tracking. The experiments attempt to recover a stroke written on a chalk-board based on the acoustic emissions the chalk makes during writing. This toy human computer interface was inspired by the tap-based interfaces of [10] and [17]. Part of this work was done concurrently to the tracking research performed in [19]. These previous works sensed vibration in a surface. This experimental setup senses the acoustic emission through the air.

4.3.1 Setup

Figure 4-12 is a picture of the chalk-board and microphone array. Six microphones are placed on a metal frame around the edge of the chalk-board. The geometry of the array is given in Table 4.1. Two microphones are placed on each of the left, right and bottom bars of the frame. These electret microphones are biased, and the resulting signals are amplified, then sampled at 96 kHz by an off-the-shelf sound-card. The five microphone pairs with the smallest aperture are used to calculate TDoAs. Their apertures are given in Table 4.2. Each sketch is drawn in the box whose chalk outline is visible in Figure 4-12(a), and whose corners are approximately $(0.3, 0.48)$ meters and $(0.68, 0.76)$ meters in the coordinates of the microphone geometry.



(a) The chalk-board and microphone array. (b) A single microphone and mount.

Figure 4-12: The experimental apparatus. Six microphones line the edges of a chalk board. The experiments consist of short, single-stroke sketches being drawn on the board.

Cross correlations were computed from a 2048 sample window ($T_W \approx 21.3$ ms). The advance between TDoA estimates is a constant 512 samples ($t_k - t_{k-1} = T_a \approx 5.3$ ms). The frequency content of the chalk's acoustic emissions depends on many factors, including the speed of the stroke, the shape of the edge of the chalk, the density of the chalk, and position of contact with the board. However, we observe that a majority of the chalk's acoustic energy lies between 1 kHz and 10 kHz. For these reasons, we use an approximate PHAT filter combined with a 1 kHz to 10 kHz bandpass filter for the Generalized Cross Correlations.

First-Order Smoothing is performed using the bounded velocity dynamics with $v_{\max} = 1\frac{\text{m}}{\text{s}}$. For this smoothing, the TDoA was quantized to the sample level (i.e. to $\frac{1}{96{,}000}$ s). The quantization levels for each pair are also given in Table 4.2. We contrast the trajectories generated by this smoothing to those generated by the GCC and a 33-tap non-causal median filter applied to the GCC's TDoA estimates. Positions are rendered from TDoA by the Nelder-Mead optimization described in Section 4.1, constrained to the bounding box of the microphone array geometry.

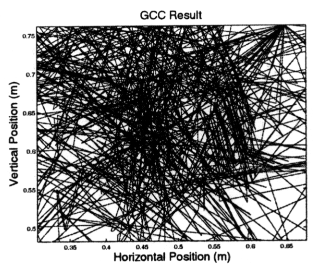
4.3.2 Results

Figures 4-13,4-15, 4-16, and 4-17 provide representative tracking comparisons, demonstrating different features of the methods being compared. A photograph of the true stroke is presented alongside the estimated trajectories from the GCC, median filter, and First-Order Smoothing. Also plotted are the results of applying the inertial Kalman filter, described in Section 4.1.1, with hand-tuned process and measurement noise covariances. The same tracking and Kalman filtering parameters were used for all of these figures.

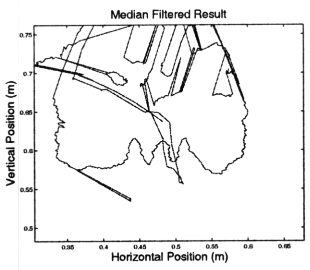
Figure 4-13 provides an example where the GCC provides poor estimates for the majority of the stroke. Figure 4-14 plots consecutive Generalized Cross Correlations, and First-Order Smoothing Posteriors for a single microphone pair observing this stroke, along with estimates rendered by the methods being compared. In Figure 4-14(a) we see a pattern of "salt" noise — different aliases dominant at different times. The median filter rejects much of this aliasing, as shown in Figure 4-14(b), but fails momentarily at around 1.5 seconds. This example demonstrates the GCC having heavy tails, and the median filter exaggerating the error due to the primary spatial aliases, as our simulations predicted in Figure 4-4 on Page 60.

By contrast, First-Order Smoothing rejects the aliases consistently. Notice, however, that First-Order smoothing fails at the sharp corners in the stroke. The chalk briefly stops at these corners, and the acoustic signal disappears. In Figure 4-14(c), at around 3.5 seconds, we see the uncertainty of the TDoA increase, as the posterior briefly widens. In Figure 4-13(c) we see that the position estimates have increased variance at the corners of these sharp transitions. The Kalman filter helps smooth these corners. Figure 4-15 provides an example where this uncertainty does not arise for every corner. In this example, we also see that the Kalman filter's smoothing can undesirably distort and remove sharp edges. Notice that the median filter does not display this added variance at sharp corners. It does, however, significantly distort the overall shape of the stroke in these regions.

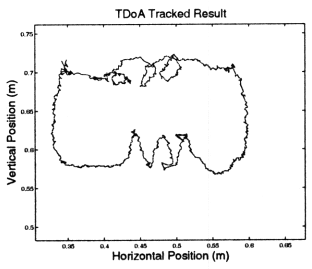
By contrast, Figure 4-16 provides an example where the GCC provides slightly



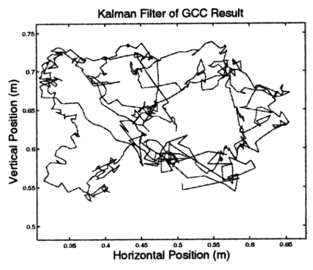
(a) GCC Estimates



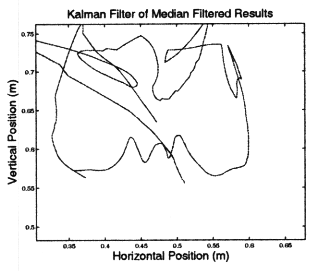
(b) Median Filter Estimates



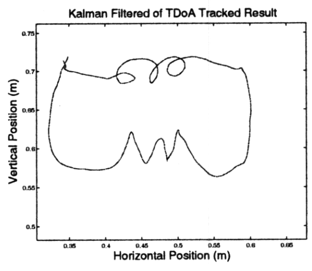
(c) First-Order Smoothing Estimates



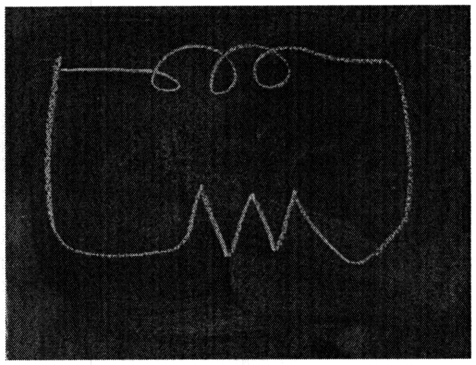
(d) Kalman Filter Applied to GCC Estimates



(e) Kalman Filter Applied to Median Filter Estimates

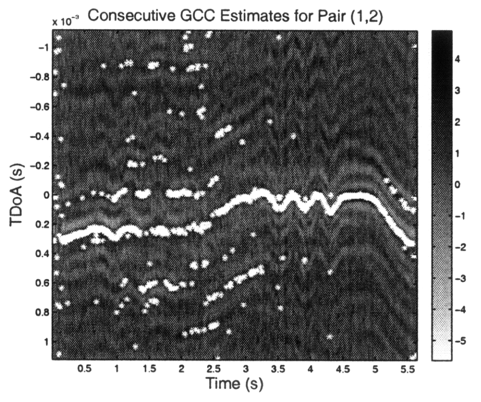


(f) Kalman Filter Applied to First-Order Smoothing Estimates

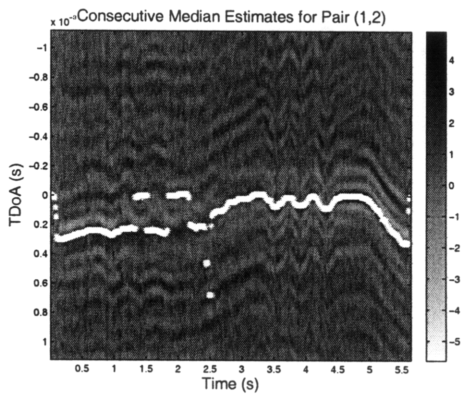


(g) Photograph of Stroke (0.37×0.28 m)

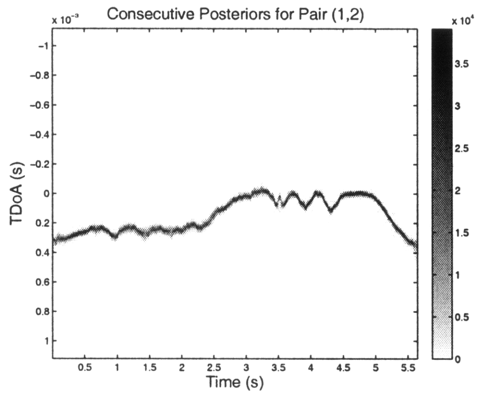
Figure 4-13: A comparison of tracking results for a single stroke (duration 5.67 seconds). The Kalman filter does not recover a meaningful trajectory from the raw GCC estimates. Even after a median filter preprocessing, spatial aliasing seriously distorts the resulting estimates. First Order Smoothing performs well both with smooth motions, and abrupt changes in direction.



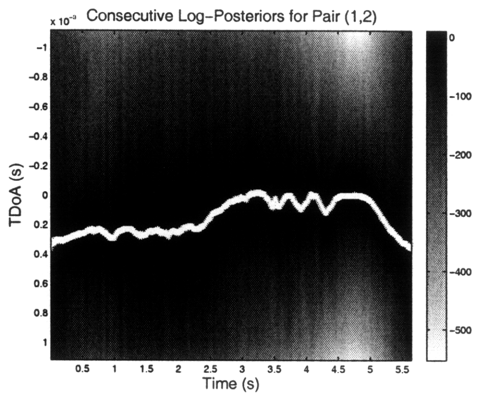
(a) GCC Estimates



(b) Median Filter Estimates

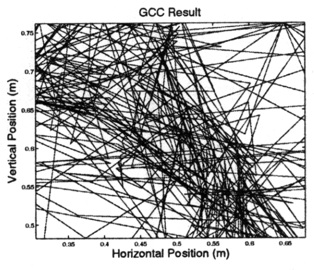


(c) First-Order Smoothing Posteriors

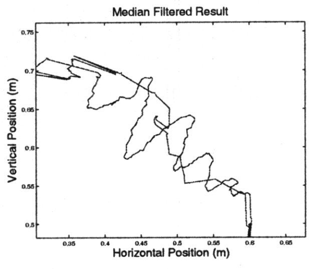


(d) First-Order Smoothing Log-Posteriors and Estimates

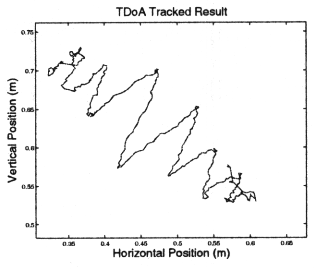
Figure 4-14: A comparison of the TDoA estimates rendered from microphone pair (1,2), during the stroke in Figure 4-13. Figures 4-14(a) and 4-14(b) overlay the GCC and median filter estimates respectively on top of an image plotting consecutive GCCs. Figure 4-14(c) plots the posterior distributions of First-Order Smoothing. Figure 4-14(d) plots the log-probabilities of Figure 4-14(c) for easier comparison to the consecutive GCCs.



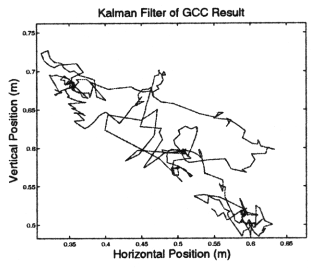
(a) GCC Estimates



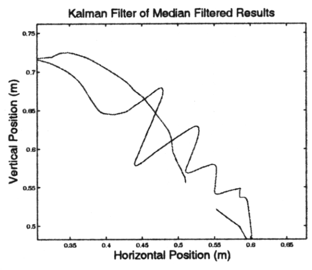
(b) Median Filter Estimates



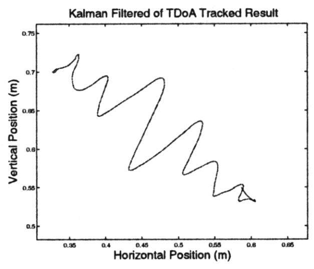
(c) First-Order Smoothing Estimates



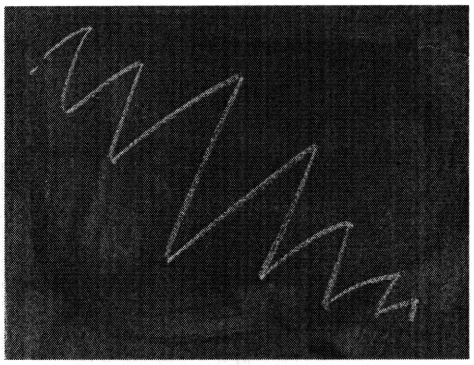
(d) Kalman Filter Applied to GCC Estimates



(e) Kalman Filter Applied to Median Filter Estimates



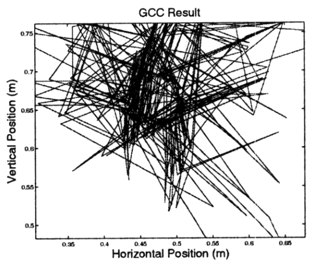
(f) Kalman Filter Applied to First-Order Smoothing Estimates



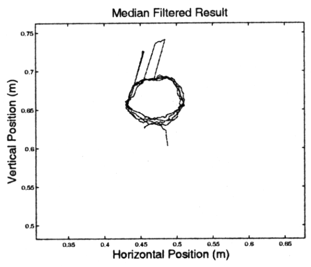
(g) Photograph of Stroke (0.37 × 0.28 m)

Figure 4-15: A comparison of tracking results for a single stroke (duration 2.66 seconds). The Kalman filter applied to the GCC estimates fails to recover the stroke, and the median filter fails on the earliest portion of the trajectory.

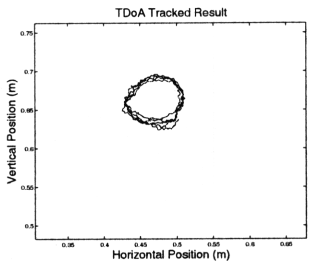
more reasonable estimates; the rough shape of the figure is visible, and can almost be recovered by the Kalman filter. Again, the median filter fails momentarily. Notice also that the Kalman filter exaggerates the radius of these stroke. The ad-hoc "inertial" model leads to this distortion. Figure 4-17 provides another example where the Kalman filter exaggerates features of the stroke. Here the last two loops are momentarily tangent, where they are not in the true stroke, or pre-Kalman filter estimates.



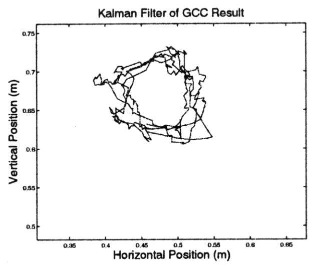
(a) GCC Estimates



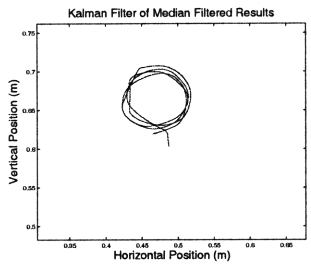
(b) Median Filter Estimates



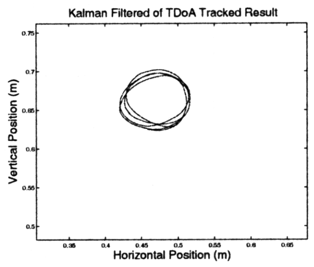
(c) First-Order Smoothing Estimates



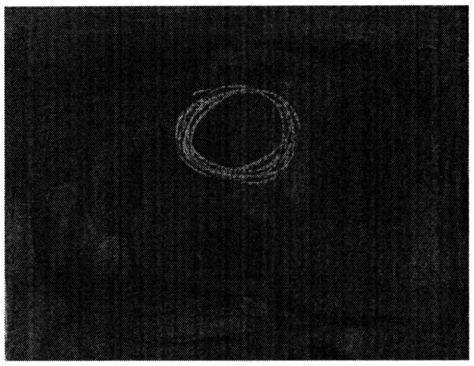
(d) Kalman Filter Applied to GCC Estimates



(e) Kalman Filter Applied to Median Filter Estimates

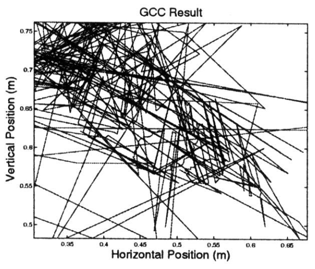


(f) Kalman Filter Applied to First-Order Smoothing Estimates

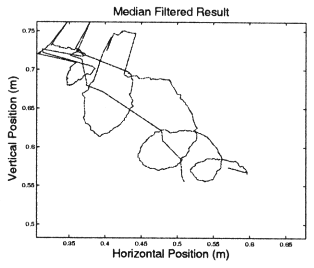


(g) Photograph of Stroke (0.37 × 0.28 m)

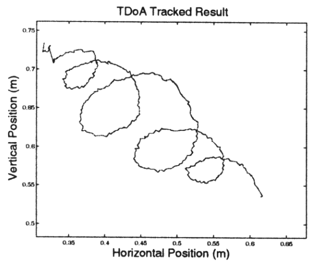
Figure 4-16: A comparison of tracking results for a single stroke (duration 5.01 seconds). The Kalman filter recovers some of the general shape from this stroke, and the median filter preprocessing performs well for all but a short interval of estimates. First-Order Smoothing performs well for the entire trajectory. In the median filter plots, the outlying estimate toward the bottom right corner is due to a lack of signal at the beginning of the recording.



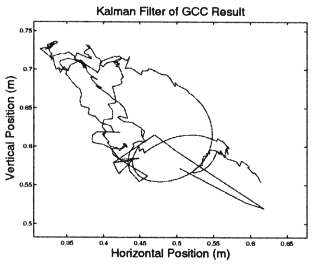
(a) GCC Estimates



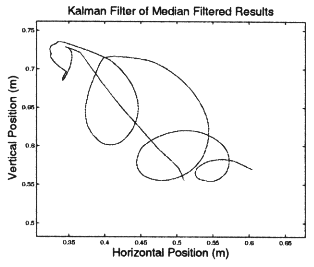
(b) Median Filter Estimates



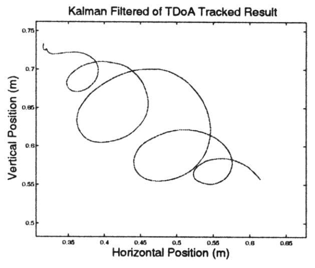
(c) First-Order Smoothing Estimates



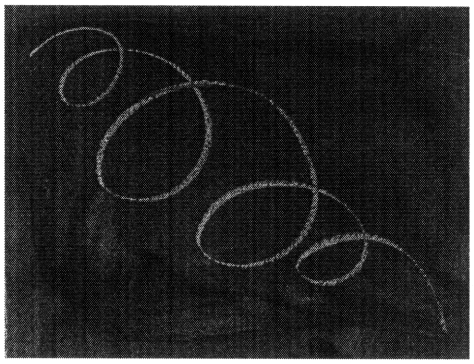
(d) Kalman Filter Applied to GCC Estimates



(e) Kalman Filter Applied to Median Filter Estimates



(f) Kalman Filter Applied to First-Order Smoothing Estimates



(g) Photograph of Stroke (0.37×0.28 m)

Figure 4-17: A comparison of tracking results for a single chalk stroke (duration 2.73 seconds). The median filter performs well when the target remains toward the center of the tracking area. However, First-Order Smoothing provides more consistent results. In both cases, the inertial Kalman filter exaggerates the loops significantly.

Chapter 5

Contributions and Future Work

This work revisits the improvement of narrow-band passive acoustic tracking by individually tracking the time difference of arrival for each pair of sensors in an array. I present a new model for TDoA evolution based on bounding the velocity of the target being tracked. This model, combined with first-order Markov assumptions, allows for an efficient form of exact marginalization with an asymptotic runtime of $O(Q \lg(Q))$, where the TDoA is quantized to one of Q levels (see Section 3.3).

I compare through simulation the TDoA and position estimates rendered by this TDoA tracking to the estimates generated by the Generalized Cross Correlation and a median filter preprocessor. When the velocity bound is accurate, this TDoA tracking provides greatly improved performance for low SNRs (see Section 4.2). Error histograms support that this performance gain was due to the rejection of spatial aliases, and the experimental data agrees with these histograms. For a range of low SNRs, TDoA tracking out-performs a Kalman Filter of these alternative position estimates, even when this filter is given the true error mean and covariance. When the velocity bound is frequently violated, the performance of this TDoA tracking method degrades. However, the quality of position estimates degrades substantially less than that of the TDoA estimates.

I provide real-world verification of TDoA tracking via an experimental apparatus (see Section 4.3). Using a six microphone array, TDoA tracking legibly recovered strokes on a chalk-board. Thought quantitative ground-truth is not available, TDoA tracking provides a serious qualitative improvement over the GCC, median filter pre-processing and Kalman filtering. The experimental results also demonstrate that TDoA tracking composes well with Kalman filtering.

TDoA tracking provides improved tracking performance using parallel computation which relies only on local information. As a result, these methods hold great promise for application to sensor networks. Sensor networks have the requisite parallelism, and benefit from algorithms which use only local information due to the communication-rate limitations. TDoA tracking could facilitate the use of sensor nodes with larger apertures, or sensor nodes with single microphones and dynamically selected pairs, as in [23].

Future work will explore the relationship between array geometry and robustness to violation of velocity bounds. In sensor networks, this information could be used to refine sensor selection. I will also explore combining the bounded velocity model of this work with the constant gain structure of Bethel and Rahikka [3] to allow for looser restrictions on TDoA evolution. Finally, the development of efficient, second-order Markov models has shown promise in early investigation.

Appendix A

Simulation and Experiment Code

This Appendix contains the Matlab® code used for both the simulations and experiments.

band_pass_modeled_noise.m

```
% band_pass_modeled_noise(N,fs,start,stop)
%   Constructs a source signal; white-noise passed through a
%     bandpass modeling filter.
% N -- half the number of samples.
% fs -- sampling frequency.
% start -- start of pass-band.
% stop  -- stop  of pass-band.
%
% [s] = band_pass_modeled_noise(N,fs,start,stop);
% s -- [2N 1] source signal.
function [s,b,a] = band_pass_modeled_noise(N,fs,start,stop)
ps = randn(2*N,1);
[b,a] = butter(4,[start/(fs/2) stop/(fs/2)],'bandpass');
s = filter(b,a,ps);
s = N*(2*(stop-start)/fs)*s./sqrt(sum(s'*s));%sum((omegas >= 1000).*(omegas <= 3000));
end
```

compare_sdoa_smoothing_brackets.m

```
% compare_sdoa_smoothing_brackets(l_to_m, ls, pairs, z, Tw, Ta, fs, c,...
%                                 sigbands, nRs, psis, wf, medK, pTs,pDs,fbK)
% Compares several TDoA estimation techniques over a range of apertures
% and signal/noise/filter conditions. See compare_sdoa_smoothing.
%
% l_to_m -- (l -> DxM)
%   Given length scale, return positions of M microphones in D dimensional space
% ls -- 1xL A number of apertures (length scales) to compare.
% pairs -- 2xP
%   Indicies of microphones to compare (each elt. in (1..M), no column
%     with identical numbers.
% z -- KxD
%   Trajectory to track.
% Tw -- Periodogram window duration (s).
%  Ta -- Advance between windows (s).
%  fs -- sampling rate (Hz)
%  c  -- speed of sound
%  sigbands -- Fx2 Bandpass parameters for signal source.
%  nRs -- Fx1 Variance of noise.
%  psis -- LxF Cell of GCC spectral weight functions (see psi_ht.m)
%  wf  -- Periodgram window function (e.g. @hann).
%  medK -- Median Filter Order
%  pTs -- Transition matrix for bounded velocity dynamics.
%  fbK -- Point at which to execute partial smoothing.

% Returns:
%  shat -- LxF  For each aperture and SNR condition, the result of
%                    compare_sdoa_amoothing.
%  zgt  -- DxW    Ground Truth for z from the center of each window.
%  sgt  -- PxW    Ground Truth for SDoA from the center of each winow
%  sdoa_lis --    GCC Evolution

function [shat,sgt,zgt] = compare_sdoa_smoothing_brackets(l_to_m, ls, pairs, z, Tw, Ta, fs, c,...
                                                    sigbands, nRs, psis, wf, medK, pTs,pDs,fbK)
L = size(ls,2);
F = size(sigbands,1);

if nargin < 16
    fbK = 1;
end

shat = cell(L,F);
sgt =  cell(L,1);
zgt =  cell(L,1);

for l = 1:L
    m = l_to_m(ls(l));

    for f = 1:F
        [shat{l,f},sgt{l},zgt{l}] = compare_sdoa_smoothing(m, pairs, z, Tw, Ta, fs, c,...
                                        sigbands(f,:), nRs(f), psis{l,f}, wf, medK, ...
                                                       pTs{l},pDs{l},fbK);

    end
end

end
```

compare_sdoa_smoothing.m

```
% compare_sdoa_smoothing(m, pairs, z, Tw, Ta, fs, c,...
%                      sigband, nR, psi, wf, medK, pT,pD,fbK)
%  m -- DxM
%    Positions of M microphones in D dimensional space
%  pairs -- 2xP
%    Indicies of microphones to compare (each in (1..M), no column
%      with identical numbers.
%  z  -- KxD
%    Trajectory to track.
%  Tw -- Periodogram window duration.
%  Ta -- Advance between windows.
%  fs -- sampling rate
%  c  -- speed of sound
%  sigband -- [Wstart Wstop] Bandpass parameters for signal source
%  nR -- variance of noise
%  psi -- GCC spectral weight function (see psi_ht.m)
%  wf  -- Periodgram window function (e.g. @hann)
%  medK -- Median Filter Order
%  pT -- {P 1} cell of transition matricies for bounded velocity dynamics.

% Returns:
% There are S=6 strategies right now:
%     (1) Quantize the ground truth.
%     (2) GCC maximum.
%     (3) Median Filter of (2).
%     (4) First-Order Filtering
%     (5) First-Order Smoothing
%     (6) First-Order Partial Smoothing
%  zhat -- DxWxS  Estimates of z from the W windows for the three methods.
%  shat -- PxWxS  Estimates of the Sample difference of arrival for the
%                   three methods.
%  zgt  -- DxW    Ground Truth for z from the begining of each window.
%  sgt  -- PxW    Ground Truth for SDoA from the beginning of each winow

function [shat,sgt,zgt,sdoa_lis,pL,pP,pLP] = compare_sdoa_smoothing(m, pairs, z, Tw, Ta, fs, c,...
                                              sigband, nR, psi, wf, medK, pT,pD,fbK)
%  Isolate parameters
D = size(m,1);
M = size(m,2);
P = size(pairs,2);
K = size(z,1);
us = 1;

if nargin < 15
    fbK = 1
end

% Generate signals
s = band_pass_modeled_noise(2*K,fs,sigband(1),sigband(2));
x = positions_and_source_to_sensors(m,z,s,nR,fs,c);

sgt = (fs/c)*positions_to_ddoas(m,pairs,z');


% Compute GCC
dmax = pairs_to_distances(m,pairs);
sdoa_lis = recording_to_sdoa_log_likelihoods_ab(x,pairs,dmax,Tw,wf,wf,Ta,fs,c,us,psi);

% Sample out the Ground Truth
K = size(sdoa_lis{1},2);
advance = Ta*fs;
sgt = sgt((0:K-1)*advance+1 + advance/2,:);
zgt = z((0:K-1)*advance+1 + advance/2,:)';


% Make Sample Difference of Arrival Estimates
shat = zeros([size(sdoa_lis{1},2) P 6]);

% Quantization
shat(:,:,1) = round(sgt);
% Just GCC maximized
shat(:,:,2) = sdoa_distb_to_max(m,pairs,sdoa_lis);
% Median Filter
shat(:,:,3) = medfilt1(shat(:,:,2),medK);

[pL,pP,pLP] = first_difference_map(m,pairs,sdoa_lis,pT,c,fs*us,pD,fbK);
```

```
% Forward
shat(:,:,4) = sdoa_distb_to_max(m,pairs,pL);
% Forward-Backward
shat(:,:,5) = sdoa_distb_to_max(m,pairs,pP);
% Forward-Back the front up.
shat(:,:,6) = sdoa_distb_to_max(m,pairs,pLP);

end
```

compare_tracking_brackets.m

```
%  m -- DxM
%    Positions of M microphones in D dimensional space
%  pairs -- 2xP
%    Indicies of microphones to compare (each elt. in (1..M), no column
%      with identical numbers.
%  z  -- KxD

%  fs -- sampling rate
%  c  -- speed of sound

function [zhat] = compare_tracking_brackets(m,pairs,shat,c,fs)

L = size(shat,1);
F = size(shat,2);

zhat = cell(L,F);
for l = 1:L
    for f = 1:F
        [zhat{l,f}] = compare_tracking(m,pairs,shat{l,f},c,fs);
    end
end

end
```

compare_tracking.m

```
%  m -- DxM
%    Positions of M microphones in D dimensional space
%  pairs -- 2xP
%    Indicies of microphones to compare (each elt. in (1..M), no column
%      with identical numbers.
%  z  -- KxD

%  fs -- sampling rate
%  c  -- speed of sound

function [zhat] = compare_tracking(m,pairs,shat,c,fs)

D = size(m,1);
K = size(shat,1);
S = size(shat,3);

zhat = zeros([D K S]);

for str = 1:S
    zhat(:,:,str) = sdoas_to_positions_nm(m,pairs,shat(:,:,str),c,fs,1);
end
```

```
end
```

distance_to_samples.m

```
% Distance to Samples
%   distance_to_samples(ds,fs,c,us): ds is a vector of distances, c is the
%   speed of sound, fs is the sampling frequency, us is an upsampling
%   factor.  Returns a vector of the number of samples which will elapse in
%   the time for sound to travel the distance.
%
% works so long as units match (e.g m/s Hz m)
function sdoas = distance_to_samples(ds,fs,c,us)
    if nargin < 4
        us = 1;
    end
    sdoas = round(us.*ds*fs/c);
end
```

first_difference_map.m

```
% first_difference_map(m,pairs,sdoa_lis,pT,c,fs,pD,k)
% Applies first-order filtering, smoothing and partial smoothing.
%
%
% For each pair of sensors, finds the maximizing indicies.
%  m -- DxM
%     Positions of M microphones in D dimensional space
%  pairs -- 2xP
%     Indicies of microphones to compare (each in (1..M), no column
%       with identical numbers.
%  sdoa_lis -- {P 1} cell of [Q(p) K] matricies; K is the number of
%     windows, Q(p) is the quantization level of pair p.
%
%  pT -- {P 1} cell of transition matricies for bounded velocity dynamics,
%      each [Q(p) Q(p)], where Q(p) is the number of quantization levels of
%      sensor pair P.
%  c -- speed of sound.
%  fs -- sampling frequency (Hz)
%  pD -- {P 1} cell of initial priors on TDoA, each [Q(p) 1]
%  k -- number of steps for partial smoothing.
%
%  Returns:
%      Three {P 1} cells of [Q(p) K] matricies.
%      First, the posteriors from filtering, then smoothing, then partial
%      smoothing.
%
function [pL,pP,pLP] = first_difference_map(m,pairs,sdoa_lis,pT,c,fs,pD,k)

if nargin < 8
    k = 0;
end

T = 1/fs;

M = size(m,2);
dim = size(m,1);

P = size(pairs,2);
```

```
dmax = pairs_to_distances(m, pairs);

smax = distance_to_samples(dmax,fs ,c);

if nargin  < 7
    pD = cell(1,size(pairs ,2));
    for p = 1:size(pairs ,2)
        pD{p} = ones(1+2*smax(p),1)/(1+2*smax(p));
    end
end

wins = size(sdoa_lis{1},2);

pL = cell(1,size(pairs ,2));
pPr = cell(1,size(pairs ,2)); % priors

pP = cell(1,size(pairs ,2));
pLP = cell(1,size(pairs ,2));
for p = 1:P
    pL{p} = zeros(size(sdoa_lis{p}));
    pPr{p} = zeros(size(sdoa_lis{p}));


    pLi = exp(sdoa_lis{p});

    pPr{p}(:,1) = pD{p};
    for w = 2:wins
        pPr{p}(:,w) = pT{p}*pPr{p}(:,w-1);
    end

    for w = 1:wins
     pD{p} = (pT{p}*pD{p}) .* pLi(:,w);
     pD{p} = pD{p}./sum(pD{p});
     pL{p}(:,w) = pD{p};
    end
    pP{p} = ones(size(sdoa_lis{p}));


    pP{p} = ones(size(sdoa_lis{p}));
    pP{p}(:,wins) = pPr{p}(:,wins);
    for w = wins-1:-1:1
        pP{p}(:,w) = pT{p}*(pLi(:,w+1).*pP{p}(:,w+1));
        pP{p}(:,w) = pP{p}(:,w)./sum(pP{p}(:,w));
    end
    for w = 1:wins
        pP{p}(:,w) = (pL{p}(:,w).*pP{p}(:,w))./(pPr{p}(:,w));
        pP{p}(:,w) = pP{p}(:,w)./sum(pP{p}(:,w));
    end

    pLP{p} = pL{p};

    pLP{p}(:,k) = pPr{p}(:,k);

    for w = (k-1:-1:1)
        pLP{p}(:,w) = pT{p}*(pLi(:,w+1).*pLP{p}(:,w+1));
        pLP{p}(:,w) = pLP{p}(:,w)./sum(pLP{p}(:,w));
    end
    for w = (1:k)
        pLP{p}(:,w) = (pL{p}(:,w).*pLP{p}(:,w))./(pPr{p}(:,w));
        pLP{p}(:,w) = pLP{p}(:,w)./sum(pLP{p}(:,w));
    end

end
end
```

kalman_filter.m

```
function xhatm = kalman_filter(x0,zhat,A,R,H,Q,Q0)

if nargin < 7
    Q0 = Q;
end

Pkm = Q0;
xhat = x0;

xhatm = zeros(4,length(zhat));

for i = 1:length(zhat)
K = Pkm*H'*inv(H*Pkm*H' + R);
xhat = xhat + K*(zhat(:,i) - H*xhat);
Pk = (eye(4) - K*H)*Pkm;
Pkm = A*Pk*A' + Q;
xhat = A*xhat;
xhatm(:,i) = xhat;
end


end
```

kalman_path.m

```
function [z,v,a] = kalman_path(procsig,wins,advance,fs,z0,v0)

%a = resample(randn(wins,2)*procsig,advance,1);
a = randn(1,wins,2)*procsig;
a = reshape(repmat(a,[advance 1 1]), [advance*wins 2]);
v = repmat(v0',wins*advance,1) + cumsum(a,1);
z = repmat(z0',wins*advance,1) + cumsum(v,1);

end
```

pairs_to_distances.m

```
%  Pairs to Distances
%    pairs_to_distance(ms,pairs):
%      Returns a row vector of the distances between microphone pairs.
%
%      ms is an DxM matrix of microphone locations. There are M
%      microphones located in a D dimensional space.
%
%      pairs is a 2xP matrix of microphone pairs.
%       1 <= pairs(i,j) <= M, pairs(1,j) =/= pairs(2,j).

function dmax = pairs_to_distances(ms,pairs)
    P = size(pairs,2);
    dmax = zeros(1,P);
    for p = 1:P
```

```
        dmax(p) = positions_to_doas(ms(:,pairs(1,p)),ms(:,pairs(2,p)));
    end
end
```

positions_and_source_to_sensors.m

```
% [x,w] = positions_and_source_to_sensors(m,z,s,noisesig,fs,c)
%   Generate sensor signals from geometry, trajectory etc.
%  m -- DxM
%    Positions of M microphones in D dimensional space.
%  z  -- KxD
%    Trajectory to track.
%  noisesig -- Standard Deviation of gaussian noise processes.
%  fs -- sample rate (Hz).
%  c -- speed of sound.
function [x,w] = positions_and_source_to_sensors(m,z,s,noisesig,fs,c)

M = size(m,2);
N = size(z,1);

w = noisesig*randn(N,M);

c = 340.29;
d = positions_to_doas(m,z');
tau = d/c;
sam = tau*fs;

x = s(repmat((1+N/2:3*N/2)',[1 M]) - round(sam)) + w;
end
```

positions_to_ddoas.m

```
% Positions to DDoAs
%    positions_to_ddoas(ms,pairs,pts):
%
%      ms is an DxM matrix of sensor locations.  There are M
%      sensors located in a D dimensional space.
%
%      pairs is a 2xP matrix of microphone pairs.
%       1 <= pairs(i,j) <= M,  pairs(1,j) =/= pairs(2,j).
%
%      pts is a DxT matrix of points.
%
%      Returns an TxP array of the DDoA from each point to the P different
%      pairs of sensors.
function ddoas = positions_to_ddoas(ms,pairs,pts)

    P = size(pairs,2);

    T = positions_to_doas(ms,pts);


    % distance difference of arrival
    for p = 1:P
        ddoas(:,p) = T(:,pairs(1,p)) - T(:,pairs(2,p));
    end

end
```

positions_to_doas.m

```
% Positions to DoAs
%    positions_to_doas(ms, pairs , pts ):
%
%      ms is an DxM matrix of microphone locations.  There are M
%      microphones located in a D dimensional space.
%
%      pts is a DxT matrix of points.
%
%      Returns an TxM array of the distance from each point to the M different
%      sensors.
function doas = positions_to_doas(ms,pts)
    M = size(ms,2);
    D = size(pts ,2);

    % compute distance of arrival
    doas = zeros(D,M);
    for m = 1:M
        doas(:,m) = sqrt(sum((pts - repmat(ms(:,m),1,D)).^2,1));
    end
end
```

psi_ht.m

```
%  Psi -- PHAT and Bandpass
%    psi = psi_phat_band(start , stop)
%      start is the beginning of the passband in Hz
%      stop  is the end of the passband in Hz
%
%      Returns a function psi(X,Y,fs):
%        X,Y are FxW matricies.  Each column is the result of an FFT.
%        fs is the sampling frequency of the pre-FFT signal, in Hz.
%
%        returns an FxW matrix of spectral weights.  The weights normalize
%        the magnitude of X(f,w)*Y(f,w) to one in the pass band, and are
%        zero outside the passband.
function filt = psi_ht(sigpsd ,n1psd ,n2psd ,kappa)
    if nargin < 4
        kappa = 1
    end

    function psi = filter(X,Y,fs)
        F = size(X,1);

        Gss = sigpsd(F/2+1);
        Gn1 = n1psd(F/2+1);
        Gn2 = n2psd(F/2+1);
        psiht = Gss./(Gss.*(Gn1+Gn2) + Gn1.*Gn2);
        psi = repmat(psiht , 1, size(X,2));
        norm = sqrt(sum(abs([psi ; flipud(psi(2:F/2,:))].*X.*Y).^2,1));
        %norm = max(abs(X).*abs(Y),[],1);
        %size(norm)
        psi = kappa*F*psi./repmat(norm,[1+F/2 1]);
    end
    filt = @filter;
end
```

psi_phat_band.m

```
%  Psi -- PHAT and Bandpass
%    psi = psi_phat_band(start,stop)
%      start is the beginning of the passband in Hz
%      stop  is the end of the passband in Hz
%
%      Returns a function psi(X,Y,fs):
%        X,Y are FxW matricies.  Each column is the result of an FFT.
%        fs is the sampling frequency of the pre-FFT signal, in Hz.
%
%        returns an FxW matrix of spectral weights.  The weights normalize
%        the magnitude of X(f,w)*Y(f,w) to one in the pass band, and are
%        zero outside the passband.
function filt = psi_phat_band(start,stop,kappa)
    if nargin == 2
        kappa = 1
    end
    function phi = filter(X,Y,fs)
        F = size(X,1);

        phi = kappa./(abs(X(1:F/2+1,:)).*abs(Y(1:F/2+1,:)));

        f = (0:F/2)*(fs/F);

        phi((f < start) | (f > stop),:) = 0;
    end
    filt = @filter;
end
```

recording_to_sdoa_log_likelihoods_ab.m

```
%  Recordings to SDoA Log Likelihoods A/B
%    Iterates sdoa_log_likelihoods_ab over multiple microphones pairs, but
%    with arguments in terms of distances and times instead of samples.
%
function sdoa_lis = recording_to_sdoa_log_likelihoods_ab(xs,pairs,dmax,window,wfa,wfb,...
                                                          advance,fs,c,us,filt)
    P = size(pairs,2);

    sdoa_lis = cell(P);
    if nargin < 8
        us = 1;
    end



    smax = distance_to_samples(dmax,fs,c,us);
    swin = fix(fs*window);
    if numel(swin) == 1
        swin = swin*ones(P,1);
    end
    if numel(us) == 1
        us = us*ones(P,1);
    end
    sadv = fix(advance*fs);
    for p = 1:P
        sdoa_lis{p} = sdoa_log_likelihood_ab(xs(:,pairs(1,p)),xs(:,pairs(2,p)),smax(p),swin(p),...
                                              wfa,wfb,sadv,fs,us(p),filt);
    end

end
```

recording_to_sdoa_log_likelihoods.m

```
% recording_to_sdoa_log_likelihoods(xs, pairs, dmax, window, wf, advance, fs, ...
%                                   c, us, filt)
% Invokes recording_to_sdoa_log_likelihoods_ab with the same a and b window
% function.
function sdoa_lis = recording_to_sdoa_log_likelihoods(xs, pairs, dmax, window, wf, advance, fs, c, us, filt)
    sdoa_lis = recording_to_sdoa_log_likelihoods_ab(xs, pairs, dmax, window, wf, wf, ...
                                                    advance, fs, c, us, filt);
end
```

sdoa_log_likelihood_ab.m

```
% x -- Nx1 matrix of sensor one's recording.
% y -- Nx1 matrix of sensor two's recording.
% smax -- distance between sensors in terms of samples.
% W -- duration of window in terms of samples.
% M -- advance between windows in samples.
% fs -- sampling frequency
% us -- upsampling factor to be applied.
% wfa -- window function to use on x
% wfb -- window function to use on y
% filter -- prefilter (see psi_hi.m)
%
% Returns:
% [smax K] matrix of consecutive log likelihoods (K is the number of
% windows processed).
function Li = sdoa_log_likelihood_ab(x, y, smax, W, wfa, wfb, M, fs, us, filter)
    N = length(x);

    no_wins = floor((N - W)/M);
    xwins = zeros(2*W, no_wins);
    ywins = xwins;
    apoda = repmat(wfa(W), 1, no_wins);
    apodb = repmat(wfb(W), 1, no_wins);

    ind = repmat(1+(M-1)*(1:no_wins), W, 1) + repmat((1:W)', 1, no_wins);
    xwins(W/2+1:3*W/2, :) = x(ind).*apoda;
    ywins(W/2+1:3*W/2, :) = y(ind).*apodb;

    F = size(xwins, 1);
    X = fft(xwins);
    Y = fft(ywins);
    phi = filter(X, Y, fs);

    Xz = [ phi(:,:) .* X(1:F/2+1,:); zeros((us-1)*F, no_wins); ...
           flipud(phi(2:F/2,:)) .* X(F/2+2:F,:) ];
    Yz = [ Y(1:F/2,:); zeros((us-1)*F, no_wins); Y(F/2+1:F,:) ];

    corrs = fftshift(ifft( Xz(:,:) .* conj(Yz(:,:)), 'symmetric'), 1);
    W = W*us;

    Li = (corrs((W-smax):(W+smax), :));
end
```

sdoa_log_likelihood.m

```
% sdoa_log_likelihood(x, y, smax, W, M, fs, us, wf, filter)
%   Calculate the unnormalized likelihood from a particular
%   TDoA from recordings.
% Invokes sdoa_log_likelihood_ab with the same a and b window function.
function Li = sdoa_log_likelihood(x,y,smax,W,M,fs,us,wf,filter)
    Li = sdoa_log_likelihood_ab(x,y,smax,W,W,M,fs,us,wf,filter);
end
```

sdoa_distb_to_max.m

```
% sdoa_distb_to_max(m,pairs,sdoa_lis)
% For each pair of sensors, finds the maximizing indicies.
% m -- DxM
%    Positions of M microphones in D dimensional space
% pairs -- 2xP
%    Indicies of microphones to compare (each in (1..M), no column
%      with identical numbers.
% sdoa_lis -- {P 1} cell of [Q(p) K] matricies; K is the number of
%    windows, Q(p) is the quantization level of pair p.
%
% Returns
% [K P] array of the indicies which achieve the maximums in sdoa_lis.
function sdhat = sdoa_distb_to_max(m,pairs,sdoa_lis)

P = size(pairs,2);

sdhat = zeros([size(sdoa_lis{1},2) P]);

for p = 1:P
    [a,nhat] = max(sdoa_lis{p});
    sdhat(:,p) = (nhat - ceil(size(sdoa_lis{p},1)/2));
end

end
```

sdoa_velocity_transition.m

```
function P = sdoa_velocity_transition(d, vmax, T, fs, c,us)
    smax = distance_to_samples(d,fs,c,us);
    V  = distance_to_samples(2*vmax*T, fs, c,us);

    TT = smax * 2 + 1;

    P = sparse(zeros(TT,TT));
    P = repmat(-smax:smax,[TT 1]);
    P = P - repmat((-smax:smax)',[1 TT]);
    P = sparse(abs(P) < V);
    P = P ./ repmat(sum(P,1),[TT 1]);
end
```

sdoa_velocity_transitions.m

```
function M = sdoa_velocity_transitions(ms, pairs, vmax, T, fs, c,us)

    P = size(pairs,2);

    if nargin < 7
        us = 1;
    end

    if numel(us) == 1
```

```
        us = us*ones(P,1);
    end

    M = cell(P);
    dmax = pairs_to_distances(ms, pairs);
    for p = 1:P
       M{p} = sdoa_velocity_transition(dmax(p), vmax, T, fs, c,us(p));
    end
end
```

sdoas_to_positions_nm.m

```
% DDoAs to Position - Nelder-Mead
%     ddoas_to_positions_nm(ms, pairs, dhat)
%        Calculates approximate positions from DDoA estimates using
%        fminsearch.
%
%        ms is an DxM matrix of microphone locations.  There are M
%        microphones located in a D dimensional space.
%
%        pairs is a 2xP matrix of microphone pairs.
%         1 <= pairs(i,j) <= M, pairs(1,j) =/= pairs(2,j).
%
%        dhat is a PxD matrix of DDoA estimates. dhat(p,:) are the estimates
%        for pairs(:,p).
%
function zhat = sdoas_to_positions_nm(ms,pairs,shat,c,fs,B)
%     shat = sdoa_distb_to_max(ms,pairs, sdoa_lis);

    if nargin < 6
        B = 10;
    end

    dim = size(ms,1);
    M = size(ms,2);
    D = size(shat,1);
    P = size(pairs,2);
    zhat = zeros(dim,P);

    dhat = samples_to_distance(shat,fs,c);

    d = 0;
    ds = 0;

    box = [min(ms(1,:)) min(ms(2,:)); max(ms(1,:)) max(ms(2,:))];
    center = mean(box,1);
    span = box(2,:) - center;
    lb = [center - B*span]';
    ub = [center + B*span]';
    ind = repmat([1;2],[1 M]);

    function w = objective(p)
        doas = sqrt(sum((ms - p(ind)).^2,1));
        ddoas = doas(pairs(1,:)) - doas(pairs(2,:));
        w = sum((ds - ddoas).^2);
        %if max(abs(p)) > 10*max(abs(ms(1:prod(size(ms)))))
        %     w = w + exp(1 + (10*max(abs(ms(1:prod(size(ms))))) - max(abs(p))));
        %end
        if sum(p < lb | p > ub)
            w = Inf;
        end
    end
    for d = 1:D
        ds = dhat(d,:);
        zhat(:,d) = fminsearch(@objective,center',optimset('MaxFunEvals',1000));
    end

end
```

circuit-1.m

```
in_m = 0.0254;
m = [0 15 ; 40 15 ; 23.5 1; 12.5 1; 40 30 ; 0 30]'*in_m;
pairs = [ 1 6; 4 1; 3 4; 2 3 ; 5 2]';

D = size(m,1);
M = size(m,2);
P = size(pairs,2);

fs = 96000;
advance = 512;
win = 2048;
Tw = win/fs;
Ta = advance/fs;

clear x;

xbnd = [12 12+11*4/3]*in_m;
ybnd = [19 30]*in_m;
xbndp = xbnd;
ybndp = ybnd;
%xbndp = 1.1*(xbnd(2) - xbnd(1))/2*[-0.5 0.5] + (xbnd(2) + xbnd(1))/2;
%ybndp = 1.1*(ybnd(2) - ybnd(1))/2*[-0.5 0.5] + (ybnd(2) + ybnd(1))/2;

output = 'circuit';
stem = 'rec/final-circuit--';
for i = 1:M
    x(:,i) = wavread([stem int2str(i)]);
end
%if output == 'face' / 'circles'
%    x = x(200*advance:size(x,1)-200*advance,:);
%end
%if output == 'jags'
%    x = x(1:size(x,1)-30*advance,:);
%end

%x = x(40000:size(x,1),:);



c = 340.29;
wf = @hann;
psi = psi_phat_band(1000,10000,40);
%psi = psi_band(1000,10000,4);

dmax = pairs_to_distances(m,pairs);
sdoa_lis = recording_to_sdoa_log_likelihoods_ab(x,pairs,dmax,Tw,wf,wf,Ta,fs,c,1,psi);
% GCC
shat = sdoa_distb_to_max(m,pairs,sdoa_lis);

zhat = sdoas_to_positions_nm(m,pairs,shat,c,fs,1);
standardize_figure(figure(1))
```

```
plot(zhat(1,:),zhat(2,:))
xlim(xbndp)
ylim(ybndp)
title('GCC_Result','FontSize',18)
xlabel('Horizontal_Position_(m)','FontSize',18)
ylabel('Vertical_Position_(m)','FontSize',18)
print('-dpdf',['thesis_final/figures/experiment/gcc-' output '.pdf']);

% Median

shatmed = medfilt1(shat,33);
zhatmed = sdoas_to_positions_nm(m,pairs,shatmed,c,fs,1);
plot(zhatmed(1,:),zhatmed(2,:))
xlim(xbndp)
ylim(ybndp)
title('Median_Filtered_Result','FontSize',18)
xlabel('Horizontal_Position_(m)','FontSize',18)
ylabel('Vertical_Position_(m)','FontSize',18)
print('-dpdf',['thesis_final/figures/experiment/med-' output '.pdf']);

% Probabalistic

vmax = 1;

pT = sdoa_velocity_transitions(m, pairs, vmax, Ta, fs, c);
pD = cell(size(pT,1),1);
for p = 1:P
  pD{p} = sum(pT{p} > 0,1)'./sum(sum(pT{p} > 0,1),2);
end

[pL,pP,pLP] = first_difference_map(m,pairs,sdoa_lis,pT,c,fs,pD,9);

shat2 = sdoa_distb_to_max(m,pairs,pP);
zhat2 = sdoas_to_positions_nm(m,pairs,shat2,c,fs,1);

standardize_figure(figure(1));
plot(zhat2(1,:),zhat2(2,:))
xlim(xbndp)
ylim(ybndp)
title('TDoA_Tracked_Result','FontSize',18)
xlabel('Horizontal_Position_(m)','FontSize',18)
ylabel('Vertical_Position_(m)','FontSize',18)
print('-dpdf',['thesis_final/figures/experiment/tdoa-' output '.pdf']);
% Kalman Filter

procsig = 1/5e8;
Q0 = diag([0.5,0.5,0,0]);

G = [0.5*Ta^2 0 ; 0 0.5*Ta^2 ; Ta 0 ; 0 Ta];
H = [ 1 0 0 0 ; 0 1 0 0];
A = eye(4) + [0 0 Ta 0 ; 0 0 0 Ta; zeros(2,4)];
Q = G*[ procsig*win 0 ; 0 procsig*win]*G';

R = [5e-5 0 ; 0 5e-5]*(0.5*Ta^2);

kssgcc = kalman_filter(zeros(size(m,1)*2,1),zhat(:,:),A,R,H,Q,Q0);
kss = kalman_filter(zeros(size(m,1)*2,1),zhat2(:,:),A,R,H,Q,Q0);
kssmed = kalman_filter(zeros(size(m,1)*2,1),zhatmed(:,:),A,R,H,Q,Q0);
standardize_figure(figure(2));
plot(kss(1,:),kss(2,:))
xlim(xbndp)
```

```
ylim(ybndp)
title('Kalman_Filtered_of_TDoA_Tracked_Result','FontSize',18)
xlabel('Horizontal_Position_(m)','FontSize',18)
ylabel('Vertical_Position_(m)','FontSize',18)
print('-dpdf',['thesis_final/figures/experiment/kalman-' output '.pdf']);

standardize_figure(figure(2));
plot(kssmed(1,:),kssmed(2,:))
xlim(xbndp)
ylim(ybndp)
title('Kalman_Filter_of_Median_Filtered_Results','FontSize',18)
xlabel('Horizontal_Position_(m)','FontSize',18)
ylabel('Vertical_Position_(m)','FontSize',18)
print('-dpdf',['thesis_final/figures/experiment/kalman-med-' output '.pdf']);

standardize_figure(figure(2));
plot(kssgcc(1,:),kssgcc(2,:))
xlim(xbndp)
ylim(ybndp)
title('Kalman_Filter_of_GCC_Result','FontSize',18)
xlabel('Horizontal_Position_(m)','FontSize',18)
ylabel('Vertical_Position_(m)','FontSize',18)
print('-dpdf',['thesis_final/figures/experiment/kalman-gcc-' output '.pdf']);
```

job.m

```
function [zgtsave,sgtsave,zhatsave,shatsave] = job(I,seed)
['starting_job' datestr(now)]

if nargin < 1
I = 1;
end

if nargin < 2
seed = sum(100*clock);
end

v = version;
if v(1:3) == '7.7'
    RandStream.setDefaultStream(RandStream('mt19937ar','seed',seed));
else
rand('twister',seed);
randn('seed',seed);
end




% Parameters
fs = 96000;
advance = 2048;
win = 2048;
wins = 51;
Tw = win/fs;
Ta = advance/fs;

c = 340.29;

d = 1/sqrt(2);
l_to_m = @(l) [l-d -d; -d l-d; d-l d ; d d-l; l-d d; -d d-l; d-l -d; d l-d]';
```

```
pairs = [1 2; 3 4; 5 6; 7 8]';
pairs = [2 1; 4 3; 6 5; 8 7]';
P = size(pairs,2);
D = 2;
% Path Generation
procsig = 1/4e9;
Q0 = diag([0.5,0.5,0,0]);

G = [0.5*Ta^2  0 ; 0 0.5*Ta^2 ; Ta 0 ; 0 Ta];
H = [ 1 0 0 0 ; 0 1 0 0];
A = eye(4) + [0 0 Ta 0 ; 0 0 0 Ta; zeros(2,4)];
Q = G*[ procsig*win 0 ; 0 procsig*win]*G';

R = [5e-5 0 ; 0 5e-5]*(0.5*Ta^2);

% Filter Parameters
wf = @hann;
psi = psi_band(1000,10000,4);

% Smoothing Parameters
vmax = 1;
medK = floor(1 + 4*(Tw/Ta));
medK = 9;



M = 16;

%pairs = [M/2+(1:M/2);1:M/2 ];
% alternate pairs to avoid bias in TDoA error
pairs = [[1:M/4; M/2 + (1:M/4)] [M*3/4 + (1:M/4); M/4 + (1:M/4)]];
P = size(pairs,2);

ls = [0.4 0.4];

L = length(ls);

% Brackets
fstart = 1000;
nRs = logspace(-1,2,15);
bands = 500*ones(size(nRs))'*[0 1];
sigbands = fstart*ones(size(bands)) + bands;

%nRs = repmat(nRs,[1 L]);
sigbands = [500 1000];
sigbands = reshape(repmat(sigbands',[15 1]),[2 15])';


F = size(sigbands,1);
%nRs = [6  6  6 6];

%bands = 5*logspace(2,3,5)'*[0 1];
%nRs = 0.1*ones(size(bands));
%sigbands = fstart*ones(size(bands)) + bands;

%psi = psi_phat_band(1000,10000,1);



psis = cell(L,F);
for f = 1:F
```

```
    [s,b,a] = band_pass_modeled_noise(1,fs,sigbands(f,1),sigbands(f,2));

    psis{2,f} = psi_ht(@(W) abs(freqz(b,a,W)).^2,...
                    @(W) nRs(f)^2*ones(W,1),@(W) nRs(f)^2*ones(W,1),1/4);

    psis{1,f} = psi_phat_band(100,2000,40);

end

% CONSTANTS

QNT = 1;
GCC = 2;
MED = 3;
FDT = 4;
FDS = 5;
FDTS = 6;
S = FDTS;

pTu = cell(L,1);
pDu = cell(L,1);


% pTu{2} = pTkf{1};
% pDu{2} = pDkf{1};
%pTu{2} = pTu{1};
%pDu{2} = pDu{1};
%pTu{3} = pTu{1};
%pDu{3} = pDu{1};


% Run the simulations
zgtsave = zeros(I,D,wins-1);
sgtsave = zeros(I,wins-1,P);
zhatsave = zeros(L,F,I,D,wins-1,S);
shatsave = zeros(L,F,I,wins-1,P,S);

for iter = 1:I
    m0 = 2*(rand(2,M/2) - 0.5);
    theta = 2*pi*rand(1,M/2); r = ls(1) + 0.5*rand(1,M/2);
    md = [r.*cos(theta) ; r.*sin(theta)];

    m = [m0 m0+md];

    for l = 1:L
    pTu{l} = sdoa_velocity_transitions(m, pairs, vmax, Ta, fs, c);
    pDu{l} = cell(size(pTu{l},1),1);
    for p = 1:P
        pDu{l}{p} = sum(pTu{l}{p} > 0,1)'./sum(sum(pTu{l}{p} > 0,1),2);
    end
    end

    l_to_m = @(l) m;

    % start
    x0 = Q0*rand(4,1);
    % path
    [z,v,a] = kalman_path(procsig,wins,advance,fs,x0((1:2)'),x0((3:4)'));
    % signals and TDoA estimates
    [shat,sgt,zgt] = compare_sdoa_smoothing_brackets(l_to_m, ls, pairs, z, Tw, Ta, fs, c, ...
                                                    sigbands, nRs, psis, wf, medK, pTu,pDu,9);
    % position estimates
    [zhat] = compare_tracking_brackets(m,pairs,shat,c,fs);

    zgtsave(iter,:,:) = zgt{1};
    sgtsave(iter,:,:) = sgt{1};
    for l = 1:L
        for f = 1:F
            zhatsave(l,f,iter,:,:,:) = zhat{l,f};
            shatsave(l,f,iter,:,:,:) = shat{l,f};
        end
    end
end
['done_job!_' datestr(now)]
end
```

Bibliography

[1] M.R. Allen and L.A. King. An adaptive two stage Kalman structure for passive undersea tracking. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(1):3–9, Jan 1988.

[2] Joshua N. Ash and Randolph L. Moses. Acoustic time delay estimation and sensor network self-localization: Experimental results. *The Journal of the Acoustical Society of America*, 118(2):841–850, 2005.

[3] R.E. Bethel and R.G. Rahikka. An optimum first-order time delay tracker. *Aerospace and Electronic Systems, IEEE Transactions on*, AES-23(6):718–725, Nov. 1987.

[4] Jingdong Chen, Jacob Benesty, and Yiteng Huang. Time delay estimation in room acoustic environments: an overview. *EURASIP J. Appl. Signal Process.*, (1):170–170, January 2006.

[5] Joseph Hector DiBiase. *A High-Accuracy, Low-Latency Technique for Talker Localization in Reverberant Environments Using Microphone Arrays.* PhD thesis, Brown University, Provdence, Rhode Island, March 2000.

[6] 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. *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, 1:I–121–I–124, April 2007.

[7] Arnaud Doucet, Nando De Freitas, and Neil Gordon, editors. *Sequential Monte Carlo methods in practice.* Birkhäuser, 2001.

[8] M.D. Gillette and H.F. Silverman. A linear closed-form algorithm for source localization from time-differences of arrival. *Signal Processing Letters, IEEE*, 15:1–4, 2008.

[9] F. J. Harris. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[10] Hiroshi Ishii, Craig Wisneski, Julian Orbanes, Ben Chun, and Joe Paradiso. Pingpongplus: design of an athletic-tangible interface for computer-supported cooperative play. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 394–401, New York, NY, USA, 1999. ACM.

[11] L.M. Kaplan. Local node selection for localization in a distributed sensor network. *Aerospace and Electronic Systems, IEEE Transactions on*, 42(1):136–146, Jan. 2006.

[12] C. Knapp and G. Carter. The generalized correlation method for estimation of time delay. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 24(4):320–327, Aug 1976.

[13] Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998.

[14] Juan Liu, James Reich, and Feng Zhao. Collaborative in-network processing for target tracking. *EURASIP J. Appl. Signal Process.*, 2003(1):378–391, 2003.

[15] Michael I. Mandel and Daniel P. W. Ellis. EM Localization and Separation using Interaural Level and Phase Cues. *Applications of Signal Processing to Audio and Acoustics, 2007 IEEE Workshop on*, pages 275–278, Oct. 2007.

[16] Michael I. Mandel and Daniel P.W. Ellis. A probability model for interaural phase difference. In Bhiksha Raj, Daniel P. W. Ellis, Paris Smaragdis, and Malcolm Slaney, editors, *Workshop on Statistical and Perceptual Audio Processing SAPA2006*, Pittsburgh, PA, October 2006.

[17] J.A. Paradiso, Che King Leo, N. Checka, and Kaijen Hsiao. Passive acoustic sensing for tracking knocks atop large interactive displays. *Sensors, 2002. Proceedings of IEEE*, 1:521–527 vol.1, 2002.

[18] S.S. Reddi. An exact solution to range computation with time delay information for arbitrary array geometries. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 41(1):485–, Jan 1993.

[19] G. De Sanctis, D. Rovetta, A. Sarti, G. Scarparo, and S. Tubaro. Localization of Tactile Interactions Through TDOA Analysis: Geometric vs. Inversion-Based Method. Proceedings of EUSIPCO-06, Florence, Italy, September 2006, 2006.

[20] J. Smith and J. Abel. The spherical interpolation method of source localization. *Oceanic Engineering, IEEE Journal of*, 12(1):246–252, Jan 1987.

[21] Tai-Lai Tung, Kung Yao, C. W. Reed, Ralph E. Hudson, Da-Ching Chen, and James Chen. Source localization and time delay estimation using constrained least-squares and best-path smoothing. In Franklin T. Luk, editor, *Advanced Signal Processing Algorithms, Architectures, and Implementations IX*. SPIE, Jul 1999.

[22] J. Vermaak and A. Blake. Nonlinear filtering for speaker tracking in noisy and reverberant environments. *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, 5:3021–3024 vol.5, 2001.

[23] M. Walpola, Hao Zhu, and Jinsong Zhang. Self organization algorithm for unattended acoustic sensor networks in ground target tracking. *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, pages 2350–2354, March 2007.

[24] 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, Nov. 2003.

[25] D.B. Ward and R.C. Williamson. Particle filter beamforming for acoustic source localization in a reverberant environment. *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on*, 2:1777–1780, 2002.

[26] A. Weiss and E. Weinstein. Fundamental limitations in passive time delay estimation–Part I: Narrow-band systems. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 31(2):472–486, Apr 1983.

[27] Ying Yu and H.F. Silverman. An improved TDOA-based location estimation algorithm for large aperture microphone arrays. *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, 4:iv–77–iv–80 vol.4, May 2004.

[28] D.N. Zotkin and R. Duraiswami. Accelerated speech source localization via a hierarchical search of steered response power. *Speech and Audio Processing, IEEE Transactions on*, 12(5):499–508, Sept. 2004.