Algorithms for Affective and Ubiquitous Sensing Systems and for Protein Structure Prediction

by

Na Yang

Submitted in Partial Fulfillment of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor Wendi B. Heinzelman

Department of Electrical and Computer Engineering

Arts, Sciences and Engineering

Edmund A. Hajim School of Engineering and Applied Sciences

University of Rochester

Rochester, New York

2015

Biographical Sketch

The author was born in Beijing, China. She received her Bachelor of Science degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2009, and her Master of Science degree from the University of Rochester, Rochester, NY, in 2011, both in Electrical Engineering. She began doctoral studies in Electrical Engineering at the University of Rochester in 2011 under the direction of Professor Wendi Heinzelman. Her research interests lie primarily in the interdisciplinary area of signal processing and machine learning towards affective computing applications and the broad human-computer interaction area. She interned at Microsoft Research, Redmond, WA, in 2011, and Dell Research, Santa Clara, CA, in 2014.

The following publications were a result of work conducted during doctoral study:

- Na Yang, He Ba, Weiyang Cai, Ilker Demirkol, and Wendi Heinzelman, "BaNa: A Noise Resilient Fundamental Frequency Detection Algorithm for Speech and Music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol.22, no.12, pp.1833-1848, December 2014.

- Na Yang, Jianbo Yuan, Yun Zhou, Ilker Demirkol, Wendi Heinzelman, and Melissa Sturge-Apple, "Analysis of Multiclass SVM with Thresholding Fusion for Speech-based Emotion Classification," submitted to *Computer Speech and Language (Elsevier)*.

- Na Yang, Jianbo Yuan, Yun Zhou, Ilker Demirkol, Wendi Heinzelman, and Melissa Sturge-Apple, "How Does Noise Impact Speech-based Emotion Classification?" (position paper), *Designing Speech and Language Interactions Workshop, the ACM CHI Conference on Human Factors in Computing Systems*, April 2014.

- Na Yang, Rajani Muraleedharan, JoHannah Kohl, Ilker Demirkol, Wendi Heinzelman, and Melissa Sturge-Apple, "Speech-based Emotion Classification Using Multiclass SVM with Hybrid Kernel and Thresholding Fusion," in *proceedings of the 4th IEEE Workshop on Spoken Language Technology (SLT)*, December 2012.

- He Ba, Na Yang, Ilker Demirkol, and Wendi Heinzelman, "BaNa: A Hybrid Approach for Noise Resilient Pitch Detection," in *proceedings of the IEEE Statistical Signal Processing Workshop (SSP)*, August 2012.

- Na Yang, Ilker Demirkol, and Wendi Heinzelman, "Cross-layer Energy Optimization Under Image Quality Constraints for Wireless Image Transmissions," in *proceedings of the 8th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC)*, August 2012.

- Na Yang, Ilker Demirkol, and Wendi Heinzelman, "Motion Sensor and Camera Placement Design for In-home Video Monitoring Systems," in *proceedings of IEEE Global Communications Conference (GLOBECOM)*, December 2011.

Acknowledgements

The present dissertation is a testimony to Professor Wendi Heinzelman's invaluable guidance and support over the years. I would like to thank her for giving me the opportunity to pursue research in the exciting field of affective computing and wireless communications. I feel very fortunate to have a great advisor as Professor Wendi Heinzelman that makes my Ph.D. study a memorable experience. She is a role model for my school years, and also the years after.

I would also like to express my sincere thanks to Professors Zhiyao Duan, Ehsan Hoque, Jiebo Luo and Dr. Jiye (Harry) Shi for acting as members of my thesis committee. Over the years at the University of Rochester, I have had the pleasure to work with exceptional colleagues in the Wireless Communications and Networking Group (WCNG). Among them, I would like to specifically acknowledge Ilker Demirkol, Rajani Muraleedharan, He Ba, Jianbo Yuan, Yun Zhou, Weiyang Cai, Kenneth Imade, Carmen Cortazar, Nancy Vargas, Emre Eskimez, Li Chen, Ou Yang, Tianqi Wang, and visiting student Thomas Horta for their help and friendship. I have benefited from collaboration with everyone in the laboratory. I would like to thank Professor Melissa Sturge-Apple for the psychology insights that she has provided. I am also grateful to Professor Michael Huang for admitting me into the University of Rochester, and providing help and advice for my study in a new country.

I would like to especially express my gratitude to my family for their love, support, and encouragement.

Abstract

Rapid development in sensing technologies has facilitated increased design of more affective and ubiquitous sensing environments for humans. Through affective sensing of human emotions and behaviors, devices can respond accordingly to provide the users with a better human-computer interaction experience. While affective sensing provides electronic devices with a better understanding of humans, ubiquitous sensing provides humans with a better knowledge of their environments. Wireless sensor networks (WSNs) have been proposed for different ubiquitous sensing scenarios over the decades, and in-home monitoring is one of the successful examples that have been widely deployed. Algorithms designed to optimize such in-home sensor networks can also be mapped to other domains. In particular, the problem of optimizing coverage in directional sensor networks can be mapped to the problem of predicting the structure of proteins, an important challenge for bioinformatics research that is needed for effective drug therapy design. In this dissertation, we contribute algorithms to enable affective and ubiquitous sensing systems as well as algorithms to improve protein structure prediction.

Accurate acquisition and interpretation of human physical signals are two essential components for affective sensing. In the first part of this dissertation, we develop a speech-based emotion classification system, which uses several one-against-all support vector machines with a threshold-based fusion mechanism to combine the individual outputs. A thorough performance evaluation of this system is provided for different test scenarios, including classification using noisy speech samples and samples from real users. Results show that the system achieves a six-emotion decision-level correct classification rate of 80% for an acted dataset with clean speech. Applications for this proposed emotion sensing system range from behavior studies to context-aware electronics design.

Fundamental frequency (F_0) is one of the speech features used for emotion classification. However, noise is inevitably included during the speech signal's acquisition. We present a novel noise resilient F_0 detection algorithm named BaNa that combines the approaches of harmonic ratios and Cepstrum analysis. We test the performance of the proposed BaNa algorithm using real human speech samples corrupted by different types of noise. Results show that for almost all types of noise and signal-to-noise ratio values investigated, BaNa achieves the lowest Gross Pitch Error rate among all the classic and state-of-the-art algorithms.

In the second part of this dissertation, we study the aforementioned in-home monitoring problem, considering energy efficiency for both the monitoring and transmission processes. In particular, we evaluate the performance of different camera and motion sensor placement strategies, and formulate optimization problems to achieve the minimum energy consumption, longest network lifetime, or the lowest monetary cost. In the image transmission process, we present an energy-efficient cross-layer image transmission model that allows the user to specify an image quality constraint by optimizing the lower layer parameters. Evaluations show that our scheme outperforms the default settings of the investigated commercial devices.

Inspired by the camera placement problem in WSNs, in the third part of this dissertation, we conduct an interdisciplinary study that combines the research fields of structural bioinformatics and communications to provide a novel solution to the problem of protein side chain prediction, which offers information critical to pharmaceutical research, such as structure-based drug discovery and rational drug design. We explore the application of sensor placement optimization strategies developed for WSNs applied to protein side chain prediction. The covered sensing areas in WSNs are represented by the three dimensional space occupied by atoms both on the backbone and on side chains of proteins. Rotamer preference and spatial density are taken into consideration when optimizing the atom placement

in the three-dimensional space. Our preliminary benchmark results show that the proposed algorithm can effectively reduce the number of atom collisions compared to an initialized predicted structure.

Contributors and Funding Sources

This work was supported by a dissertation committee consisting of Professor Wendi Heinzelman (advisor) and Professor Zhiyao Duan of the Department of Electrical and Computer Engineering and Professor Ehsan Hoque of the Computer Science Department and Dr. Jiye (Harry) Shi from UCB Pharma. The committee was chaired by Professor Jiebo Luo from the Computer Science Department. The following chapters of this dissertation proposal were jointly produced, and were funded by multiple sources. My participation and contributions to the research as well as funding sources are as follows.

I am the primary author of all the chapters. For Chapter 2, I collaborated with He Ba, Weiyang Cai, Dr. Ilker Demirkol, Dr. Wendi Heinzelman, and Dr. Melissa Sturge-Apple. Thomás Horta performed the initial implementation of BaNa as an Android app. The work described in this chapter was published in IEEE/ACM Transactions on Audio, Speech and Language Processing in December 2014. Part of this work was published in Proceedings of the IEEE Statistical Signal Processing Workshop (SSP) in 2012. This work was supported by the Eunice Kennedy Shriver National Institute of Child Health and Human Development Grant R01HD060789.

For Chapter 3, I collaborated with Jianbo Yuan, Yun Zhou, Dr. Ilker Demirkol, Dr. Zhiyao Duan, Dr. Wendi Heinzelman, and Dr. Melissa Sturge-Apple. The work described in this chapter was submitted to Computer Speech and Language (Elsevier) in February 2015. Part of this work was published in Proceedings of the IEEE Workshop on Spoken Language Technology (SLT) in 2012. This work was supported by the Eunice Kennedy Shriver National Institute of Child Health and Human Development Grant R01HD060789.

For Chapter 4, I collaborated with Research Project Manager Dr. Arjmand Samuel during an internship at Microsoft Research, WA, in the summer of 2011.

For Chapter 5, I collaborated with Dr. Ilker Demirkol and Dr. Wendi Heinzel-

man. The work described in this chapter was published in Proceedings of the IEEE International Wireless Communications and Mobile Computing Conference (IWCMC) in 2012. This work was supported by funding from the National Institute of Nursing Research Grant R21 NR010857-01.

For Chapter 6, I collaborated with Dr. Ilker Demirkol and Dr. Wendi Heinzelman. The work described in this chapter was published in Proceedings of the IEEE Global Communications Conference (GLOBECOM) in 2011. This work was supported by funding from the National Institute of Nursing Research Grant R21 NR010857-01.

For Chapter 7, I collaborated with Jingwei Guo, Dr. Jiye (Harry) Shi, and Dr. Wendi Heinzelman. This work was funded in part by UCB Pharma, and by the Center for Emerging and Innovative Sciences (CEIS), an Empire State Development-designated Center for Advanced Technology.

Table of Contents

List of Tables

List of Figures

1 Introduction

In this dissertation, we describe our work developing algorithms for affective sensing systems via speech-based emotion sensing, ubiquitous sensing systems via in-home video monitoring, and applying sensing algorithms to protein structure prediction.

1.1 Affective Sensing Systems via Speech-based Emotion Sensing

1.1.1 Motivation

Affective sensing is the field of study concerned with recognizing and interpreting human emotions by capturing the user's physical state, behavior, or physiological data. Emotion can be used for behavior studies, monitoring of people with mental health concerns, and context-aware system design. Various affective sensors have been developed to sense people's emotions using speech prosody [6], facial affects [12], body gestures [13], or physiological signals such as blood volume pulse and galvanic skin response [14]. Emotion sensors interpret the emotional state of humans and adapt some behaviour accordingly, giving an appropriate response

for those emotions.

Speech contains rich information for effectively conveying emotions in the communication between humans, and this has motivated researchers to explore the area of emotion classification based on speech. A speech-based emotion classification system consists of feature extraction from speech signals and emotion classification based on these extracted features using machine learning algorithms.

1.1.2 Challenges

Although speech-based emotion classification has been studied over the years, mining useful emotion information solely from speech features is still a challenging task. The emotion classification method should be designed to be able to accurately capture speech features in different noisy environments. Therefore, robust feature extraction methods are needed. Also, the noise-resilience of the proposed system should be evaluated using noisy speech data as well as speech from real users.

Since speech in real-life interactions may be emotionally ambiguous, we are motivated to introduce a pre-set confidence threshold to determine whether to classify the emotion for one speech sample or reject it.

Privacy issues must be considered when deploying applications using users' emotions. Therefore, when we design the speech-based emotion classification system, we only analyze speech features instead of the speech content.

1.1.3 Contributions

The contributions of Part I of the dissertation are:

- We develop a noise resilient fundamental frequency (F_0) detection algorithm named BaNa. We provide an extensive study of the performance of BaNa

along with 7 other classic and state-of-the-art F_0 detection algorithms for a range of noisy speech and music signals with different types of noise and noise levels [15] [16].

- We propose a speech-based six-class emotion classification system using several one-against-all support vector machines with a threshold-based fusion mechanism to combine the individual outputs. A thorough performance evaluation of this system is provided for different test scenarios, including classifications on noisy speech data and tests using a dataset with non-professional acted emotions [17] [18] [19].

- We implement a mobile emotion sensor on a Windows Phone platform called 'Listen-n-feel' for a binary emotion classification for happy or not happy.

1.2 Ubiquitous Sensing Systems via In-home Video Monitoring

1.2.1 Motivation

In-home video monitoring systems have been used for applications ranging from surveillance and intruder detection to elderly and infant monitoring. This research was initially inspired by a psychological study on parent-teen relationships. Interparental relationship and parenting plays an important role in child development [20]. Parenting deficits may result in teen problems such as low self-esteem, poor peer-relationships, or poor school functioning. Due to the easy deployment and the large volume of family activities, the in-home environment is ideal for conducting parent-teen interaction observations or experiments. Movements and gesture interactions of family members can be captured by multiple video cameras placed in certain rooms throughout the home.

1.2.2 Challenges

Collecting and analyzing user data in real parent-teen interactions are not easy. The major challenges and concerns to build such an in-home video monitoring system are listed as follows.

- In-home monitoring should be collected unobtrusively, in order to minimize any interference that may be introduced to the parent-teen relationship experiments. Therefore, cameras and other monitoring sensors are preferred to be battery-powered and transmit the data wirelessly. To conduct continuous monitoring, the lifetime of the monitoring system should be maximized. Therefore, it is necessary to design an energy-efficient data acquisition and transmission system using different techniques, such as sensor placement optimization and parameter optimization.

- Activities of participants cannot be constrained in the experiments. For example, family members can go to any places they want to in the room, as they normally do in their real life. Given this assumption, the unpredictable movements of participants causes a coverage requirement for the the video surveillance system design, which can be achieved by placement optimization of surveillance cameras.

1.2.3 Contributions

The contributions of Part II of the dissertation are:

- Development of an image transmission model that allows the user to specify an image quality constraint. Lower layer parameters are optimized, such as transmit power and packet length, to minimize the energy dissipation in image transmission over a given distance [21].

- Optimization of motion sensor and camera placement for in-home monitoring system design, in order to achieve the minimum energy consumption, longest network lifetime, or lowest monetary cost [22].

1.3 Application of Sensing Algorithms to Protein Structure Prediction

1.3.1 Motivation

In the pharmaceutical and biotechnology industries, protein structure prediction is important for many applications, such as novel enzyme design and antibody therapeutics. Widely-used experimental techniques for protein structure determination include X-ray diffraction crystallography and nuclear magnetic resonance spectroscopy. However, experimental derivation of the atomic structures is both time-consuming and costly [23] [24]. We are motivated to explore the application of sensor placement optimization strategies developed for wireless sensor networks (WSNs) applied to protein side chain prediction. The covered sensing areas in WSNs are represented by the three dimensional space occupied by atoms of proteins.

1.3.2 Challenges

A major difficulty in protein structure prediction is the excessive computational complexity caused by an extremely large sampling space. The number of optimization parameters and the number of constraints needed to determine the optimum protein structure grow exponentially with the number of residues in a protein. A small group of residues each with a few rotation degrees of freedom already yields a huge number of possible side chain conformations. Therefore, an

efficient sampling strategy is highly desired to explore the large search space, such that a near native protein structure can be found in a reasonable amount of time with high accuracy and confidence.

1.3.3 Contribution

The contribution of Part III of the dissertation is to conduct an innovative interdisciplinary study that combines the research fields of structural bioinformatics and communications to provide effective solutions to the problem of backbone-dependent protein side chain prediction.

1.4 Dissertation Organization

These aforementioned system development challenges form the motivation and objectives of designing an affective sensing system via speech-based emotion sensing, a ubiquitous sensing system via in-home video monitoring, and applying sensing algorithms to protein structure prediction. Fig. 1.1 shows the main requirements for designing such systems.

The organization of the rest of this dissertation is as follows. In Chapter 2, we propose a noise resilient hybrid F_0 detection algorithm named BaNa for speech and music. In Chapter 3, we present a speech-based emotion classification method with thresholding fusion for an enhanced classification performance. A full analysis is provided for different test scenarios. In Chapter 4, a mobile emotion sensor named 'Listen-n-feel' is proposed for mobile Windows platforms and supports classifications for happy or not happy emotions. In Chapter 5, we investigate cross-layer energy minimization given an image quality constraint for wireless image transmissions. In Chapter 6, a motion sensor and camera deployment optimization method is presented, and issues of coverage, energy consumption,

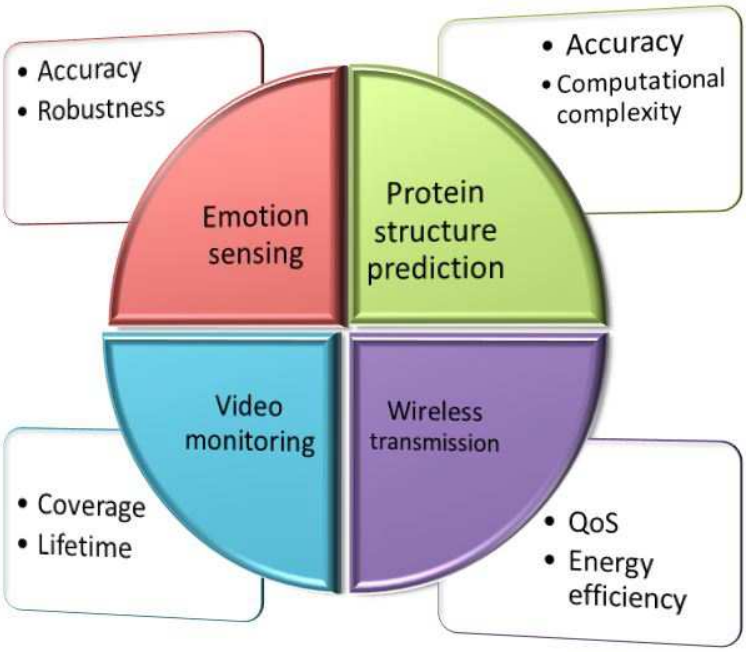


Figure 1.1: Main requirements for designing an affective sensing system via speech-based emotion sensing, a ubiquitous sensing system via in-home video monitoring, and applying sensing algorithms to protein structure prediction.

network lifetime, and monetary cost are investigated. In Chapter 7, we map the coverage optimization concept in directional sensor networks to the protein side chain structure prediction problem, and we present preliminary prediction accuracies using a proposed algorithm that considers the preferred angles of the protein side chains as well as the need to solve atom collisions within the protein structure.

Part I

Speech-based Emotion Classification

2 BaNa: A Noise Resilient Fundamental Frequency Detection Algorithm for Speech and Music

2.1 Introduction

For human speech, pitch is defined by the relative highness or lowness of a tone as perceived by the human ear, and is caused by vibrations of the vocal cords. Since pitch is a subjective term, in this chapter we use the objective term fundamental frequency (F_0), which is an estimate of pitch. If there were perfectly periodic speech signals, F_0 would be the inverse of the period of voiced speech. However, the interference of formant structure for speech signals, or the interference of spectral envelope structure for music signals, makes the accurate detection of F_0 difficult. Also, due to the aperiodicity of the glottal vibration itself and the movement of the vocal tract that filters the source signal, human speech is not perfectly periodic [25]. Additionally, accurate F_0 detection is difficult when the speech signal is corrupted with noise. Therefore, F_0 detection has always been a challenge in speech signal analysis.

A variety of speech-based applications can benefit from a more precise and robust F_0 detection algorithm. For example, F_0 detection is essential in automatic speech recognition, where pitch-accent patterns can be used to improve recognition performance [26], or homophones can be differentiated by recognizing tones [27]. For synthesized speech to be natural and intelligible, it is crucial to have a proper F_0 contour that is compatible with linguistic information such as lexical accent (or stress) and phrasing in the input text. Therefore, F_0 modeling can be used for speech synthesis [28] [29]. F_0 and azimuth cues can be jointly used for speech localization and segregation in reverberant environments [30]. Moreover, in emotion detection or other affective measurement, it has been found that prosodic variations in speech are closely related to one's emotional state, and the F_0 information is important for the identification of this state [31]. A warning system has been developed in [32] to detect if a driver exhibits anger or aggressive emotions while driving, using statistics of the F_0 value and other metrics. Some health care providers and researchers implemented F_0 detectors and other behavior sensing technologies on mobile devices, such as smartphones, for behavioral studies [5] [33] or patient monitoring, such as the clinical trials conducted by the University of Pittsburgh for detecting depression in patients [34]. However, for these types of applications, the vehicle noise captured by the detector or the ambient noise captured by mobile devices may strongly influence the F_0 detection performance.

F_0 detection also plays a very important role in music signal analysis and music information retrieval, and has a broad range of applications. Music notation programs use F_0 detection to automatically transcribe real performances into scores [35]. Reliable F_0 extraction from humming is critical for query-by-humming music retrieval systems to work well [36]. Music fingerprinting technology also uses F_0 information for music identification among millions of music tracks [37]. F_0 detection in noisy music is also challenging. Music may be recorded in noisy en-

vironments such as in a bar or on the street. Noise may also be introduced by the recording device itself. One challenge is that the F_0 generated from tonal musical instruments spans a large range, normally from 50 Hz to 4,000 Hz [38]. For musical signals with high F_0, the wide range of possible F_0 candidates increases the likelihood of finding a wrong F_0 value. The other challenge is that, unlike for human speech, the sound for musical signals can last for several seconds, thus the overlapped musical tones can also be considered as noise. Due to these reasons, when performing F_0 detection in real scenarios, the quality of the input signal may be greatly degraded. Therefore, F_0 detection of musical signals in noisy environments is necessary.

Adding noise may introduce spectral peaks in the spectrum of the speech signal or distort the shape of the speech peaks, depending on the type and level of the noise. The key to detecting F_0 from noisy speech or music is to differentiate speech or music spectral peaks from noise peaks. In Fig. 2.1, we plot the spectrum of one frame from a clean speech file and the same frame with babble noise at 0 dB SNR. By examining this frame, we can see that F_0 is located at 192 Hz. By comparing the spectrum of the clean speech and the noisy speech, we can see that the added noise peaks distort the shape of the speech peaks, causing the highest point of the peak to be shifted. For the noise at 0 dB SNR, the amplitudes of the noise peaks can even be comparable with the amplitudes of the speech peaks. However, their locations in the frequency domain are not periodic, and the distribution of the noise peaks in the frequency range varies for different types of noise. Thus, the locations of the spectral peaks are affected less by the additive noise than the amplitudes of the peaks. Therefore, the ratios of harmonic frequencies are essential to find F_0 from a noisy signal.

Also, as seen in the spectrum of the noisy speech shown in Fig. 2.1, the first four harmonics are located at 391 Hz, 581 Hz, 760 Hz, and 958 Hz. The spectral peak located at 485 Hz is from the noise signal. We can see that the harmonics

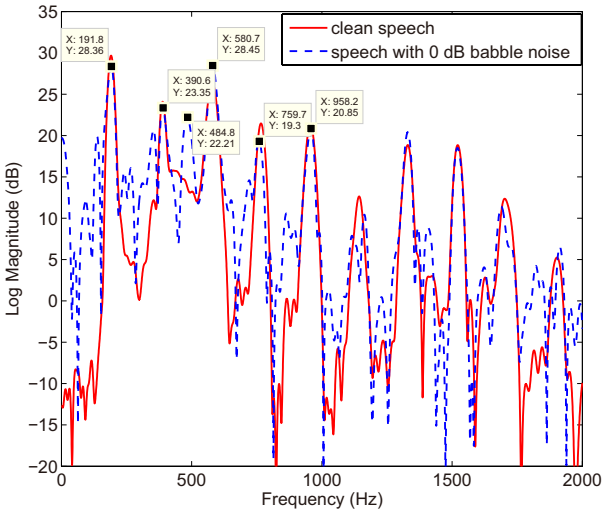


Figure 2.1: Spectrum of one frame of clean speech and speech with babble noise at 0 dB SNR.

are not exactly spaced at integer multiples of the fundamental frequency F_0 in the frequency domain, and the higher order harmonics have larger drift than the lower order harmonics. Therefore, we need to set a tolerance range to account for the drifts when calculating the ratios of harmonic frequencies.

As existing F_0 detectors, such as Cepstrum [39], HPS [40], and Praat [41], do not perform well when the input data is noisy, we are motivated to design a noise resilient F_0 detection algorithm that is better suited for practical uses. The BaNa algorithm is a hybrid F_0 detection algorithm that combines the idea of using the ratios of harmonic frequencies with tolerance ranges and the Cepstrum approach to find F_0 from a noisy signal. In this chapter, we discuss F_0 detection for both speech and music signals, and we describe the simple modifications of BaNa required for music F_0 detection. We show that using the ratios of harmonic frequencies with pre-tuned tolerance ranges for F_0 detection enables the algorithm to be resilient to additive noise. We also show that incorporating Cepstrum and post-processing techniques can improve the F_0 detection performance.

In addition, we extend the work in [42] by evaluating the BaNa algorithm on

a range of speech databases and by comparing it with seven classic and state-of-the-art F_0 detection algorithms. We test the proposed BaNa algorithm on real human speech and music samples corrupted by various types and levels of realistic noise. Evaluations show the high noise resilience of BaNa compared to the classic and state-of-the-art F_0 detection algorithms. For noisy speech at 0 dB SNR, BaNa achieve 20% to 35% Gross Pitch Error (GPE) rate for speech and 12% to 39% GPE rate for music. Also, we discuss issues with implementing BaNa on a smartphone platform. Test results on a real device show that our implementation of BaNa can process recorded speech files with a reasonable speed, opening the door for real-time F_0 detection on mobile platforms.

The rest of this chapter is organized as follows. Section 2.2 provides a brief survey of well-known F_0 detection algorithms and also some of the most recent F_0 detection algorithms. Section 2.3 describes the BaNa algorithm for F_0 detection in noisy speech. Experimental settings and extensive experimental results comparing the BaNa algorithm with several classic and state-of-the-art F_0 detection algorithms using different speech databases are presented in Section 2.4 and Section 2.5, respectively. Section 2.6 presents the slight modifications of the BaNa algorithm to improve its performance for F_0 detection in noisy music. We describe an implementation of the BaNa F_0 detection algorithm in Section 2.7. Finally, Section 2.8 concludes this chapter.

2.2 Related Work

Among the well-known classic F_0 detection algorithms, autocorrelation function (ACF) and cross correlation are among the most widely used time domain methods. A number of algorithms have been developed based on these two approaches. Average Magnitude Difference Function (AMDF) [43] is a variation of ACF, which calculates a formed difference signal between the delayed signal and the original

one. Since the AMDF algorithm does not require any multiplications, it is desirable for real-time applications. Praat [41] considers the maxima of the autocorrelation of a short segment of the sound as F_0 candidates, and chooses the best F_0 candidate for each segment by finding the least cost path through all the segments using the Viterbi algorithm. YIN [44] uses a novel difference function similar to autocorrelation to search for the F_0 period. It further refines the F_0 detection result using some post-processing methods. Two types of modified difference function used in YIN are proposed in [45]. The RAPT F_0 tracker [46], on the other hand, is a variation of cross correlation, which computes the F_0 by extracting the local maxima of the normalized cross correlation function.

In the frequency domain, F_0 is found by searching for harmonic peaks in the power spectrum. The Cepstrum method [47] [39] is among the most popular algorithms. Cepstrum is found by computing the inverse Fourier transform of the log-magnitude Fourier spectrum, which captures the period in the speech harmonics, and thus shows a peak corresponding to the period in frequency.

Schroeder's frequency histogram [40] enters all integer submultiples of all the peaks in the spectrum in a histogram. Since F_0 is the integer submultiple of all the harmonics, in an ideal case, the entry with the highest weight in Schroeder's frequency histogram is the correct F_0. As pointed out in [48], Schroeder's frequency histogram is susceptible to octave errors, as F_0 and $F_0/2$ will have the same weight in Schroeder's frequency histogram. In cases where noise peaks are selected, Schroeder's histogram will make mistakes by selecting the greatest common divisor of both the harmonics and the noise peaks.

The Harmonic Product Spectrum algorithm (HPS) [40] multiplies the original signal with downsampled signals, thus in the frequency domain, the spectra of all the downsampled signals line up the peaks at the F_0 value for isolation. Another harmonic summation method is proposed in [49], which modifies the HPS method for multiple F_0 estimation in polyphonic music. The harmonic components' energy

distribution is used, and F_0 candidates are selected using a competition mechanism. The algorithm is tested on three different instruments. However, for these harmonic summation methods, noise peaks with high amplitudes can be easily mistaken for harmonic peaks at low SNR scenarios. Since our proposed BaNa algorithm only relies on the locations of the harmonic peaks to calculate the frequency ratios of those spectral peaks, with the peak's amplitude information only being considered for peak selection, we show in Section 2.5.1 and Section 2.6.5 that the BaNa algorithm is more robust than the HPS algorithm for noisy speech and noisy music.

The PEFAC (Pitch Estimation Filter with Amplitude Compression) algorithm proposed in [50] is another frequency-domain F_0 detection algorithm for noisy speech. This approach estimates F_0 by convolving its power spectral density in the log-frequency domain with a filter that sums the energy of the F_0 harmonics while rejecting additive noise that has a smoothly varying power spectrum. Also, amplitude compression is applied before filtering to attenuate narrow-band noise components.

Some F_0 estimators operate in the time-frequency domain by applying time-domain analysis on the output of a filter bank. In the algorithm proposed by Jin and Wang [51], a new frequency channel selection method is proposed to select less corrupted channels for periodicity feature extraction. F_0 scores for each F_0 state are derived given the periodicity features and are given to a hidden Markov model for F_0 state tracking.

Recently, an increasing number of F_0 detection algorithms have been designed using a data-driven statistical approach to combat noise, such as the algorithms described in TAPS [52], Wu [53], and SAFE [54]. In [52], Huang et al. propose an F_0 estimation method that uses Temporally Accumulated Peaks Spectrum (TAPS). Since the harmonic structure of voiced speech changes more slowly than the noise spectrum over time, spectral peaks related to F_0 harmonics would stand

out after temporal accumulations. Clean and noisy speech data is required to train the peak spectrum exemplar set and the Gaussian mixture model.

The Wu algorithm [53] is also a statistical approach, which integrates a new method for extracting periodicity information across different channels, and a Hidden Markov Model for forming continuous F_0 tracks. The modeling process incorporates the statistics extracted from a corpus of natural sound sources. The SAFE algorithm [54] also uses a data-driven approach to model the noise effects on the amplitudes and locations of the peaks in the spectra of clean voiced speech.

However, these data-driven approaches may not always be practical. Since these algorithms are trained with known noise types and specific noise levels, the noise information of the test sample is also required as input to the model. However, this information is not always available, since the user often does not know the type of noise, and it is even harder to measure the noise level. The proposed BaNa algorithm, on the other hand, does not require any prior information about the noise.

Though most F_0 detection algorithms were developed for F_0 detection in speech, a number of the aforementioned algorithms have also been used in music. The YIN and Praat algorithms are evaluated in [55] for synthetic signal and real-time guitar signal F_0 tracking. In [56], F_0 detection performance of the HPS algorithm and its variation called Cepstrum-Biased HPS are compared for interactive music. Clean cello and flute pieces are used in the experiments. However, robust F_0 detection in noisy music is still a topic that needs to be explored.

In this chapter we perform an extensive quantitative comparison analysis to show the performance, in terms of Gross Pitch Error (GPE) rate, for our proposed F_0 detection algorithm, BaNa, and several of the aforementioned algorithms (YIN, HPS, Praat, Cepstrum, PEFAC, SAFE, and Wu) for noisy speech and music signals.

2.3 BaNa F_0 Detection Algorithm for Speech

In this section, we describe the BaNa hybrid F_0 detection algorithm for speech.

2.3.1 Preprocessing

Given a digital speech signal, preprocessing is performed before the extraction of the F_0 values. In the BaNa algorithm, we filter the speech signal with a bandpass filter. Let F_0^{min} and F_0^{max} denote the lower limit and upper limit for F_0 values of human speech, respectively. The lower bound of the bandpass filter is set to F_0^{min}. The upper bound is set to $p \cdot F_0^{max}$, where p is the number of spectral peaks captured that will later be used for F_0 detection.

2.3.2 Determination of the F_0 candidates

Since harmonics are regularly spaced at approximately integer multiples of F_0 in the frequency domain, we use this characteristic of speech in the proposed BaNa algorithm to achieve noise resilience. If we know the frequency of a harmonic and its ratio to F_0, then F_0 can be easily obtained. However, even if a harmonic is discovered, its ratio to F_0 is unknown. Therefore, we propose an F_0 detection algorithm that looks for the ratios of potential harmonics and finds the F_0 based on them by applying the following steps.

Step 1: Search for harmonic peaks

Spectral peaks with high amplitudes and low frequencies are preferred to be considered for F_0 candidates, since peaks with high amplitudes are less likely to be caused by noise, and peaks with low frequencies are easier to be identified to be harmonics by calculating the ratios. Peaks with high frequencies may be high order harmonics, which cannot be used to calculate harmonic ratios, since we only consider the ratios of the first p harmonics. Therefore, we consider the p peaks

with amplitudes higher than a certain threshold and with the lowest frequencies to derive F_0 candidates. We use the peak detection algorithm provided in [57] to search for the peaks in the spectrum. In [57], spectral peaks with small amplitudes are filtered out by setting a peak amplitude threshold, and peaks located very close to dominant peaks are smoothed by setting a threshold of the window width for smoothing. Settings that we use for the number of selected peaks p, the peak amplitude threshold parameter, and the window width parameter for the smoothing function in the peak detection function are presented in Table 2.2.

Let $\hat{F}_i$ and $\left|\hat{H}_i\right|$ represent the frequencies and the magnitudes of the p spectral peaks with the lowest frequencies whose magnitudes are above a certain threshold, where $i = 0, \cdots, p-1$. We place the p peaks in ascending order of frequency to obtain the set of $\hat{F}_i$, denoted as $\hat{F}$. For most human speech, energy concentrates in the low frequency part, thus some or all of the p peaks are likely to be at the first p harmonics, which are at $m \times F_0$, $m = 1, \cdots, p$. For each frame, F_0 candidates are derived from the ratios of the frequencies of $\hat{F}$ using the following algorithm.

Step 2: Calculate F_0 candidates

We calculate the ratios $R_{ij} = \hat{F}_j/\hat{F}_i$ for all $\hat{F}_i, \hat{F}_j$, where $i < j$, and $i, j = 0, \cdots, p-1$. Take the number of selected spectral peaks $p = 5$ for example. If a calculated ratio R_{ij} falls into any tolerance range of the harmonic ratios shown in the left table in Fig. 3.1, we are able to find to which harmonics $\hat{F}_i$ and $\hat{F}_j$ correspond. For harmonic ratios with small numbers, we set adjacent tolerance ranges to be bounded with each other, i.e., the upper bound of the tolerance range of the ratio of $\hat{F}_4$ and $\hat{F}_3$ is the same as the lower bound of the tolerance range of the ratio of $\hat{F}_3$ and $\hat{F}_2$, which is $(5/4 + 4/3)/2 = 1.29$, as shown in Fig. 3.1. For harmonic ratios with large numbers, the width of the tolerance range is set to 0.2 or 0.4, depending on the ratio number. We show in Section 2.4.3 that these tolerance range numbers are tuned to achieve the best F_0 detection performance.

A potential F_0 candidate can be obtained by dividing the harmonic by its ratio

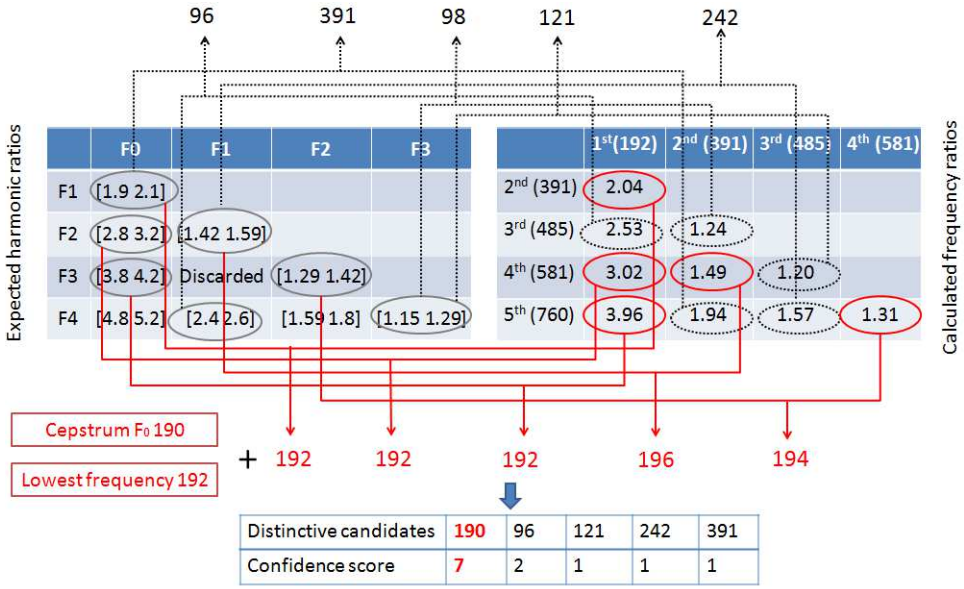


Figure 2.2: Tolerance ranges for harmonic ratios when the number p of selected spectral peaks is set to 5, and an example to illustrate the procedure for determining the F_0 candidates.

to F_0: $\tilde{F} = \hat{F}_i/m$, where $m = 1, \cdots, p$. Note that due to the imperfect periodicity of human speech, the harmonics may not be exactly on integer multiples of F_0, and we observed that higher order harmonics have even larger drift than lower order harmonics in practice. Therefore, we set a smaller ratio tolerance range for lower order harmonics, and we set a larger ratio tolerance range for higher order harmonics. In total, C_2^p ratio values are calculated between every pair of $\hat{F}$. Since both ratios of F_1/F_0 and F_3/F_1 are equal to 2, it is not trivial to differentiate to which harmonics this ratio belongs. In our algorithm, we assume it belongs to F_1/F_0 and calculate the F_0 candidate based on that.

In order to combat these octave errors, the proposed BaNa algorithm adds two other F_0 candidates. One added candidate is the spectral peak with the smallest frequency value, since we have found that in some cases only the F_0 peak is high enough to be detected. The other added F_0 candidate is the F_0 value found by the Cepstrum method. The reason is that the p spectral peaks we choose

mainly belong to low frequency values. For some rare cases, the higher order harmonics (e.g., 5th to 10th harmonics) are found to yield higher spectral peak values compared to the low order harmonics. In that case, the spectral peaks at low frequencies are more vulnerable to noise. However, since the Cepstrum method depicts the global periodicity of the spectrum, and considers all spectral peaks, it can help to detect the F_0 in those rare cases. In Section 2.5.2, we show the benefit of including the spectral peak with the smallest frequency value and the Cepstrum F_0 as additional candidates.

The number of F_0 candidates derived from the ratio analysis and the two added candidates, K, is then less than or equal to $C_2^p + 2$. Among these K F_0 candidates, the ones that are out of the F_0^{min} to F_0^{max} human speech range are discarded, and the number of candidates is reduced from K to K'. If no candidate is derived from the ratios, we set the F_0 value to 0 Hz. We then order the K' candidates in ascending order of frequency. F_0 candidates that are within ξ Hz of each other are considered to be "close" candidates. For each of these K' candidates $\tilde{F}_k$, where $k = 1, ..., K'$, we count the number of "close" candidates U_k, and select the one with the largest number of "close" candidates to be a "distinctive" candidate $\check{F}_d$, where $d = 1, ..., D$, and D is the number of "distinctive" candidates. The "distinctive" candidate and its "close" candidates are deleted from the candidate list. If there is more than one candidate that has the same largest number of "close" candidates, we select the one with the smallest frequency to be the "distinctive" candidate. We continue the same procedure for the remainder of the list until the list is empty. We set the number of "close" candidates, including the chosen candidate itself, to be the confidence score V_d for the corresponding "distinctive" candidate $\check{F}_d$. Among the D "distinctive" candidates, where $D \leq K'$, the ones with higher confidence scores are more likely to be F_0.

In Fig. 3.1, we use the frame shown in Fig. 2.1 to illustrate the process of

calculating F_0 candidates. In Fig. 2.1, the dotted line represents the spectrum of one frame of speech with 0 dB babble noise. The five selected spectral peaks that have the lowest frequencies are located at 192 Hz, 391 Hz, 485 Hz, 581 Hz, and 760 Hz. The 485 Hz peak is caused by the noise signal, and the remaining four peaks are from the speech signal. We map each calculated frequency ratio in the right table in Fig. 3.1 to one expected harmonic ratio in the left table in Fig. 3.1. For example, the ratio of the 5^{th} and 4^{th} spectral peaks is $\hat{F}_5/\hat{F}_4 = 760/581 = 1.31$, which maps to the [1.29 1.42] frequency ratio tolerance range for the expected frequency ratio of the 3^{rd} and 2^{nd} harmonics. Therefore, the F_0 candidate is derived as 581/3=194 Hz. In this example, all calculated frequency ratios are mapped to one expected harmonic ratio in the left table, which results in 10 F_0 candidates. The Cepstrum candidate and the peak with the lowest frequency are added as two additional F_0 candidates, which are 190 Hz and 192 Hz, respectively.

If we use the parameters shown in 2.4.3, all the 12 candidates are within the $F_0^{min} = 50$ Hz to $F_0^{max} = 600$ Hz range for F_0. Candidates that are within $\xi = 10$ Hz of each other are considered to be "close" candidates. In Fig. 3.1, the bottom table shows the "distinctive" candidates and their confidence scores. The 190 Hz candidate has the highest confidence score, which is very close to the ground truth F_0, i.e., 191 Hz calculated from the corresponding clean speech signal. In Fig. 3.1, correct F_0 candidates are listed on the bottom and are marked by solid red lines. Incorrect F_0 candidates are listed on the top and are marked by dotted black lines. We can see that the candidates calculated from the 485 Hz noise peak are all incorrect candidates.

2.3.3 Selection of F_0 from the candidates

In Section 2.3.2, the "distinctive" candidates of individual frames are obtained independently. However, the F_0 values of neighboring frames may correlate, since the F_0 values of human speech exhibit a slow time variation, and hence, large F_0

jumps among subsequent frames are rare. Therefore, we use the Viterbi algorithm [58] for post-processing to go through all the candidates in order to correct F_0 detection errors, similar to the post-processing used in the Praat algorithm [41]. We aim to find a path that minimizes the total cost, which consists of two parts: the frequency jumps between the candidates of two consecutive frames, and the inverse of the confidence scores of each "distinctive" candidate.

Let $\check{F}_i^n$ denote the i^{th} "distinctive" F_0 candidate of frame n and N_{frame} denote the number of frames in the given speech segment. Moreover, let p_n denote the index of the chosen F_0 candidate for the n^{th} frame. Thus, $\{p_n | 1 \leq n \leq N_{frame}\}$ defines a path through the candidates. For each path, the path cost is defined to be

$$PathCost\left(\{p_n\}\right) = \sum_{n=1}^{N_{frame}-1} Cost\left(\check{F}_i^n, \check{F}_j^{n+1}\right), \tag{2.1}$$

where $p_n = i$ and $p_{n+1} = j$. The $Cost$ is used to calculate the cost of adjacent frames. We define the function $Cost$ by using the F_0 differences between the adjacent frames and the confidence score of the candidates. The F_0 difference is modeled similarly with the *transition cost* defined in the Praat algorithm [41]. The larger the F_0 difference, the higher the $Cost$ should be. We present the F_0 difference in the Mel scale, which is a perceptual scale of F_0 judged by listeners. The perceived F_0 in the Mel scale has a logarithm relation with the F_0 in frequency, as shown in (2.2):

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right). \tag{2.2}$$

Therefore, in the cost function, the F_0 difference in frequency is modeled as the logarithm of the F_0 division in the Mel scale. The other part of the cost function is modeled using the confidence score. We assign a lower cost to those F_0 candidates with higher confidence scores, thus we use the inverse of the confidence score in the expression of the cost function. A weight w is introduced to balance the two parts. The setting for this value is shown in Table 2.2. Then, $Cost$ is defined

mathematically as

$$Cost\left(\check{F}_i^n, \check{F}_j^{n+1}\right) = \left|\log_2 \frac{\check{F}_i^n}{\check{F}_j^{n+1}}\right| + w \times \frac{1}{V_i^n}, \tag{2.3}$$

where V_i^n is the confidence score of the i^{th} "distinctive" F_0 candidate of the n^{th} frame.

We use the Viterbi algorithm to find the minimum cost path, i.e., the path that reduces the F_0 jumps the most, while giving priority to the F_0 candidates with higher confidence scores. The optimal path is found for each voiced part in the speech. The Praat algorithm also uses the Viterbi algorithm to choose F_0 from several F_0 candidates for each frame. However, the F_0 candidates of Praat are local maxima of the autocorrelation of each frame, which have the same confidence score to be selected as F_0. On the other hand, the F_0 candidates in BaNa have different confidence scores, and thus F_0 candidates derived from noise spectral peaks are less likely to be selected as F_0. Therefore, the cost function of BaNa's Viterbi algorithm shown in (2.3) is different from that in the Praat algorithm. The complete BaNa algorithm that describes the selection of the peaks and the calculation and selection of the F_0 candidates is given in Algorithm 1.

For each frame, the time complexity to calculate K F_0 candidates by calculating frequency ratios of p selected peaks is $\mathrm{O}\left(p^2\right)$. The time complexity to calculate D 'distinctive' candidates from K' remaining candidates is $\mathrm{O}\left(K'^3\right)$, which is the most complex process. The time complexity to use the Viterbi algorithm to choose the final F_0 from 'distinctive' candidates is $\mathrm{O}\left(D^2\right)$.

2.4 Experimental Settings for BaNa F_0 Detection For Speech

In this section, we present the speech and noise databases we use for F_0 detection performance evaluation, the error measurement metric, and parameter tuning of

Algorithm 1 The BaNa F_0 Detection Algorithm

// ***For frame*** n***:***
// ***Select harmonic peaks***
select $\hat{F}^n$: the p peaks with lowest frequencies
// ***Calculate*** F_0 ***candidates***
number of candidates $K \leftarrow 0$
for i =1 to p, $j = i + 1$ to p **do**
    ratio $R_{ij} = \hat{F}_j^n / \hat{F}_i^n$
    for m =1 to p, $m' = m + 1$ to p **do**
        if R_{ij} falls in the left table of Fig. 3.1 and close to $\frac{m'}{m}$ **then**
            $K \leftarrow K + 1$, $\tilde{F}_K^n \leftarrow \hat{F}_i^n / m$
        end if
    end for
end for
$K \leftarrow K + 1$, add spectral peak with the lowest frequency $\hat{F}_1^n$: $\tilde{F}_K^n \leftarrow \hat{F}_1^n$
$K \leftarrow K + 1$, add Cepstrum F_0: $\tilde{F}_K^n \leftarrow Cepstrum\ F_0$
discard $\tilde{F}^n$ that are out of the F_0^{min} to F_0^{max} range
$K' \leftarrow$ number of remaining candidates $\tilde{F}^n$
number of "distinctive" candidates $D^n \leftarrow 0$
while $\exists \tilde{F}^n \neq null$ **do**
    for k =1 to K' **do**
        if $\tilde{F}_k^n \neq null$ **then**
            num. of "close" candidates of $\tilde{F}_k^n$: $U_k \leftarrow 0$
            for $l = 1$ to K' **do**
                if $\left|\tilde{F}_l^n - \tilde{F}_k^n\right| \leq \xi$ Hz **then**
                $U_k \leftarrow U_k + 1$
                end if
            end for
        end if
    end for
    $D^n \leftarrow D^n + 1$, $V_D^n \leftarrow \max U_k$
    "distinctive" candidate $\check{F}_{D^n}^n \leftarrow \tilde{F}^n$ with max U_k
    $\tilde{F}^n$ with max $U_k \leftarrow null$
    all "close" candidates of $\tilde{F}^n$ with max $U_k \leftarrow null$
end while
// ***For all frames within a voiced segment:***
// ***Choose*** F_0 ***from "distinctive" candidates***
for n =1 to number of frames N_{frame} **do**
    for i, j =1 to D^n **do**
        $Cost\left(\check{F}_i^n, \check{F}_j^{n+1}\right) = \left|\log_2 \frac{\check{F}_i^n}{\check{F}_j^{n+1}}\right| + w \times \frac{1}{V_i^n}$
    end for
end for
return $\{p_n\}$ of $\min\{PathCost\} \leftarrow Viterbi\,(Cost)$, where path $\{p_n\}$ denotes F_0 for all frames

the proposed algorithm.

2.4.1 Speech and noise databases

Noisy speech samples can be generated by adding noise recorded in noisy environments to clean speech samples. Using this approach, the ground-truth F_0 values can be obtained from the clean speech. An alternative approach is to use speech samples directly recorded in real noisy environments, such as the SPEECON database [59], where additive noise, reverberations, and channel distortions are present. The ground-truth F_0 values in the SPEECON database are derived by manually F_0-marked recordings from a close speaking microphone with relatively little noise (clean speech). Several F_0 detection algorithms use the SPEECON database to evaluate their performance [60] [61] [62].

In this work, we use noisy speech samples generated from clean speech and different types of additive noise.

The clean speech samples we use are taken from four English speech databases: LDC [1], Arctic [63], CSTR [3], and KEELE [4]. Since female speakers normally have higher F_0 values than male speakers, approximately an equal number of speech samples from male and female speakers are chosen from these databases. Also, since the frequency characteristics in speech differ from person to person, we select speech samples from all the available speakers within these databases. Table 2.1 presents the specifications of these speech databases.

The LDC database is the Emotional Prosody Speech and Transcripts Database from Linguistic Data Consortium. It is chosen because it includes speech samples with strong emotions such as hot anger and elation, for which the F_0 values may change dramatically even within a short utterance. In the BaNa algorithm, the difference of F_0 values for neighboring frames is taken into consideration by the Viterbi algorithm. Therefore, the LDC database helps to investigate whether this

Table 2.1: Evaluated speech databases and their features. Parameters are tuned using samples from the Arctic database.

Speech databases	**Emotion**	**# of speakers**	**# of selected samples**	**% of voiced frames**	**Has F_0 ground truth?**
Arctic [63]	neutral	4	10	54.2%	No
LDC [1]	various	7	20	50.4%	No
CSTR [3]	neutral	2	100	50.3%	Yes
KEELE [4]	neutral	10	10	50.4%	Yes

discontinuity in F_0 values may influence the performance. The Arctic, CSTR and KEELE databases all contain speech samples with neutral emotion. All the speech samples used for the evaluation are included in the BaNa toolkit [64].

To test the noise resilience of the investigated algorithms, eight types of noises are added to the original signals with different SNR levels. The noise database we use is the NOISEX-92 noise database [65], in which we choose six different types of real life background noise: speech babble (labeled as *babble* in the figures for performance comparison), destroyer engine room noise (*engine*), destroyer operations room noise (*operation*), factory floor noise ($factory$), vehicle interior noise ($vehicle$), high frequency radio channel noise ($highfreq$), as well as two common types of noise: white noise ($white$) and pink noise ($pink$). To generate noisy speech with a certain SNR value, the signal energy is calculated only on the voiced part, and the noise is amplified or attenuated to a certain level to meet the target SNR value.

2.4.2 Error measurement metric

For the noisy speech data, if the detected F_0 deviates more than 10% from the ground truth value, it is counted as a gross pitch error. Otherwise, it is counted as a fine pitch error. The Praat algorithm also uses the 10% deviation range in

their error measurement in [41]. Gross Pitch Error (GPE) rate is the percentage of incorrectly detected F_0 values in voiced speech segments. GPE rate has been widely used as the error measurement metric for F_0 detection [52] [54] [66]. Mean and standard deviation of Fine Pitch Errors (FPE) are also used in this study. FPE is calculated by the relative deviation of the detected F_0 from the ground truth F_0, with the unit in percent, for any pitch that does not represent a Gross Pitch Error [67] [68].

The F_0 ground truth values for the CSTR and KEELE databases are provided, which are obtained from the simultaneously recorded laryngograph signals. We downloaded the speech data and the ground truth values for the CSTR and KEELE databases from the SAFE toolkit [69], and then shifted the ground truth values in time as needed to line up with the F_0 detected by all the algorithms tested. For the LDC and Arctic databases with no F_0 ground truth provided, we use auto-labeled F_0 values of the original clean speech as the ground truth F_0 values and the voiced/unvoiced delineation, since the original speech samples are clean and with very little background noise. To best estimate the ground truth F_0 values, we calculate the detected F_0 values of three algorithms: PEFAC, YIN and Praat, which all perform well in F_0 detection for clean speech. For one frame, if the detected F_0 values from all three algorithms are within 10%, we assume that this frame is voiced, and the auto-labeled ground truth is determined by averaging the three detected F_0 values. Otherwise, we assume that the frame is unvoiced and do not detect F_0 for that frame.

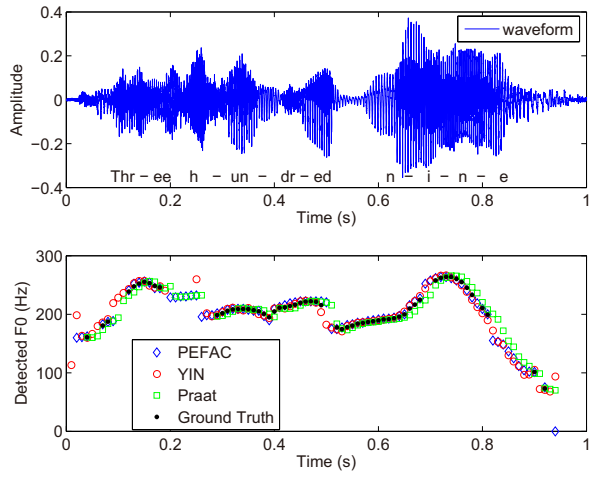
Fig. 3(a) shows an example of a clean speech recording of the utterance 'three hundred (and) nine' along with the auto-labeled F_0 values as the ground truth. The word 'and' in the middle is skipped and is not spoken. We can see that for most of this clean utterance, the detected F_0 values from the three algorithms are very close. We use black solid dots to represent the ground truth F_0 values, which are calculated by averaging the detected F_0 values from PEFAC, YIN and Praat.

We also note that the detected F_0 values from these three algorithms differ at frames corresponding to unvoiced stop consonants, i.e., 'th' in 'three' and 'h' in 'hundred', and discontinuities, i.e., the spaces between two words. Those frames are regarded as unvoiced and are ignored. For some frames, no F_0 value is shown on the plot for Praat, since Praat has its own voiced/unvoiced frame detection, and those frames are considered as unvoiced by Praat. The corresponding spectrogram is shown in Fig. 3(b), in which the lowest dark red curve verifies the calculated F_0 ground truth in Fig. 2.3a. The frame length used to compute the spectrogram is 60 ms.

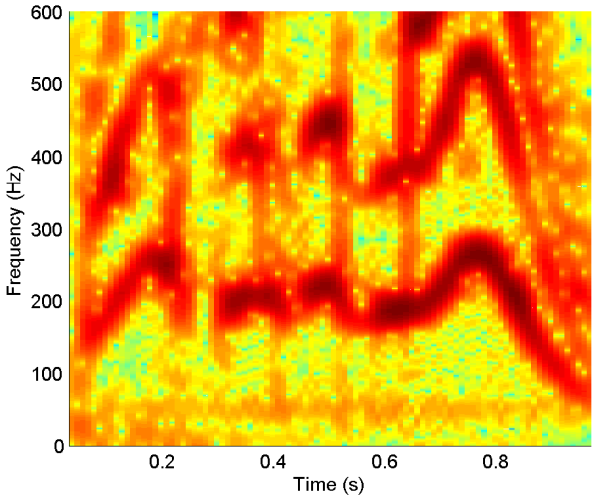
The MATLAB code for the BaNa algorithm is available on the University of Rochester Wireless Communications and Networking Group's website [64]. Although the voiced/unvoiced speech detection is not within the scope of our work, we provide one version of the MATLAB implementation of the BaNa algorithm with an automatic voice marker [64]. The voiced/unvoiced speech detector used in this version of the BaNa code is the one implemented in [47] as the voiced/unvoiced speech detector for the Cepstrum F_0 detection algorithm. Frames with a dominant cepstrum peak, with an amplitude higher than the amplitude of the second highest peak by a certain threshold, are considered as voiced frames. However, we have not evaluated the performance of this voiced/unvoiced speech detector on noisy speech. Other voiced/unvoiced speech detectors are also available in the literature [54] [60].

2.4.3 Parameter tuning

The frame shift is set to 10 ms in order to obtain smooth F_0 detection results. The absolute value of the Fourier transform of the Hann windowed speech signal is calculated, with the FFT size set to $2^{16} = 65{,}536$ to provide good frequency resolution. Candidates that are within $\xi = 10$ Hz of each other are considered to be "close" candidates. Since the F_0 of human speech is normally higher than



(a)



(b)

Figure 2.3: For one clean speech utterance: a) speech waveform and the auto-labeled ground truth F_0 derived from three algorithms: PEFAC, YIN, and Praat, and b) the spectrogram. The frame length used to compute the spectrogram is 60 ms.

50 Hz and can be as high as 600 Hz for children or female voices [70], we set the lower limit and the upper limit for F_0 of human speech to be $F_0^{min} = 50$ Hz and $F_0^{max} = 600$ Hz, respectively.

There are several parameters in the BaNa algorithm that can be pre-tuned to achieve a more accurate estimate of F_0. The Arctic samples are used for the tuning of these parameters, and the set of parameters that provides the lowest GPE rate averaged over all levels of noise and all types of the NOISEX-92 noise [65] is chosen as the parameter set used in our work.

The parameter settings are shown in Table 2.2. To obtain a stable estimate of F_0, the frame length is chosen to be at least three times the F_0 period. Since the minimum F_0 we consider for both speech and music is 50 Hz, the frame length is thus $1/50 \times 3 = 0.06$ s, i.e., 60 ms. We also list in Table 2.2 other frame length values that we have tested. Using the 20 ms frame length, which is one F_0 period at 50 Hz, results in a higher GPE rate. Although using the 90 ms frame length can slightly reduce the GPE rate, the temporal resolution is sacrificed.

Parameters in the spectral peak selection process are also tuned, including the number of spectral peaks p chosen to calculate the F_0 candidates, the spectral peak amplitude threshold and the threshold of the window width for smoothing, which is the width of the smoothing function applied before spectral peak detection. With these parameters being properly set, spectral peaks with low amplitudes and small widths are not chosen. We tested the performance of BaNa by choosing more or fewer spectral peaks, which means possibly more or fewer harmonics, but we found that choosing 5 peaks provides good F_0 detection performance. Also, choosing more spectral peaks increases the complexity in calculating the F_0 candidates.

Other parameters that are tuned are the tolerance range for the harmonic ratios used in the left table of Fig. 3.1, and the weight parameter used in the cost function in (2.3). Note that these parameters represent the optimal set across

Table 2.2: Optimal values of tuned parameters, and other values of the parameters for which BaNa algorithm is tested.

Parameters	**Optimal value**	**Other values tested**
Frame length	60 ms	20 ms, 90 ms
Number of chosen spectral peaks p	5	3, 4, 6, 7
Spectral peak amplitude threshold in peak selection	1/15 of the highest peak	1/25, 1/20, 1/10 of the highest peak
Window width for smoothing in the frequency domain in peak selection	50 Hz	40 Hz, 60 Hz, 70 Hz, 80 Hz
Tolerance range for harmonic ratios	Numbers shown in Table I	Narrowed range, extended range
Weight w in the cost function in (2.3)	0.4	0.05, 0.1, 0.2, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9

all noise types and SNR values for the Arctic speech database; they may not be optimal for a given noise type or SNR value or samples from other databases. A user could, of course, optimize the parameters for specific noise conditions, but we will show in Section 2.5 that using these tuned parameters provides good performance without the need for tuning for specific noise environments. Note that for all the other F_0 detection algorithms, we choose their default parameters in the evaluation.

To evaluate the parameter sensitivity of the BaNa algorithm on new types of noise, we use another widely-used noise database [2] with eight types of common ambient noise, including airport, babble, car, exhibition, restaurant, street, subway, and train noise. This noise database was used to construct the AURORA noisy speech database [71] for speech recognition. Note that the AURORA noise database is only used for this parameter sensitivity test. All the remaining per-

formance evaluations in this work are performed on noisy speech and noisy music generated using noise samples from the NOISEX-92 noise database [65].

We compare the performance of BaNa on the LDC database [60] by using 1) the set of parameters provided in this work, that are tuned on the Arctic database and the NOISEX-92 noise database [65], and 2) the parameter sets that are individually optimized on a specific type of noise from the the AURORA noise database [2] that yields the lowest GPE rates for the LDC database, averaged over 0 dB, 5 dB, 10 dB, 15 dB, and 20 dB SNR values.

As shown in Fig. 2.4, the difference in the performance when using the individually optimized parameter sets and when using the parameter set selected in this work is relatively small for most noise types. These results show that the performance of BaNa is not very sensitive to the specific parameters chosen. Thus, we can trade a slight drop in the GPE performance of BaNa for the benefit of not needing to optimize the parameters for a specific type of noise environment.

2.5 F_0 Detection Performance For Speech Signals

In this section, we compare the F_0 detection performance of the proposed BaNa algorithm with that of several classic and state-of-the-art algorithms on speech signals in various noisy environments and for a wide range of SNR values. Seven algorithms are considered due to their popularity or good performance: YIN, HPS, Praat, Cepstrum, PEFAC, SAFE, and Wu. These algorithms have been described in Section 2.2. The source code for YIN, Praat, Cepstrum, PEFAC, SAFE, and Wu are from [72], [73], [47], [74], [69], and [75], respectively. We implement the HPS algorithm based on the algorithm described in [40]. F_0 detection in eight different types of noise environments are evaluated, where noisy speech samples are

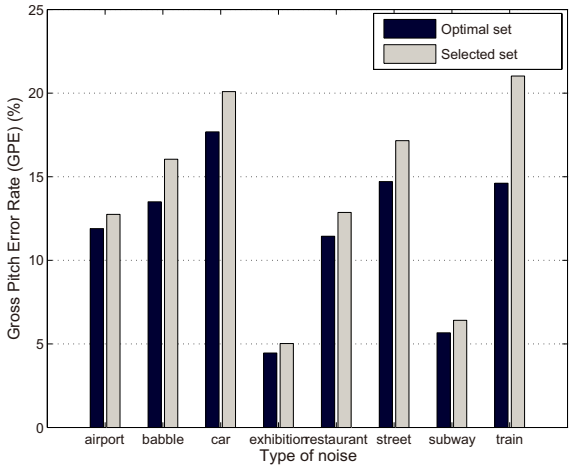


Figure 2.4: GPE rates of BaNa for the LDC database [1] with eight types of AURORA noise [2] averaged over all SNR values, using individually optimized parameter sets that provide the lowest GPE rates for a specific type of AURORA noise, and using the tuned parameter set selected in this work. Detected F_0 deviating more than 10% from ground truth are errors.

generated by adding background noise to clean real speech samples with different noise power levels to achieve different SNR values.

Note that in our study, we only detect F_0 when only one speaker is speaking or only one instrument is played. If multiple people are speaking or multiple instruments are played at the same time, multiple F_0 values coexist. Multiple F_0 detection, as studied in work such as [76] [77] [78] [79], is not within the research scope of this work.

2.5.1 F_0 detection performance for speech

We test all the F_0 detection algorithms on each one of the speech databases mentioned in Section 2.4.1, except the Arctic database, which was used for tuning the BaNa parameters. The GPE rate is evaluated as a function of SNR value, where the GPE rate is averaged over all types of noise for each SNR value.

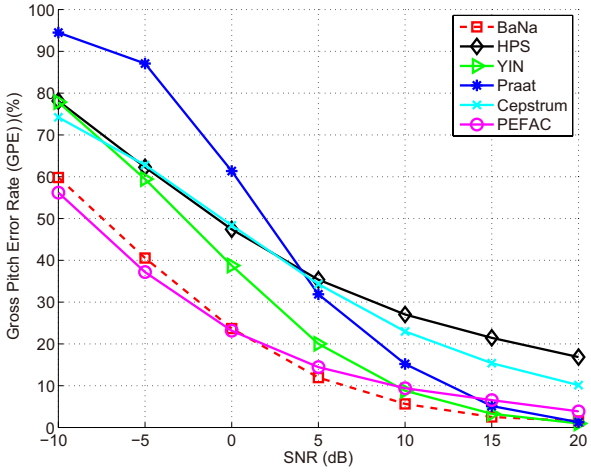


Figure 2.5: GPE rate of the different algorithms for the LDC database [1], averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

For the LDC database with emotional utterances, Fig. 2.5 depicts the results, which shows that the BaNa algorithm achieves the best F_0 detection accuracy, i.e., the lowest GPE rate, among all of the algorithms for 0 dB SNR and above 0 dB SNR. PEFAC performs slightly better than BaNa at -5 dB SNR and -10 dB SNR. BaNa achieves the lowest GPE rate of 20.6%, which is obtained by averaging over -10 dB, -5 dB, 0 dB, 5 dB, 10 dB, 15 dB, and 20 dB SNR levels. Similar to the BaNa algorithm, the HPS algorithm is also based on the ratios of the potential harmonics. However, in real speech, the harmonics are not integer multiples of F_0, which may greatly affect the F_0 detection performance. We can also see that the BaNa algorithm has a very high resilience to severe noise, as it only wrongly detects 23.7% of F_0 values with noise at 0 dB SNR.

For a more stringent evaluation, we have also tested all algorithms on the LDC database using the GPE rate with a 5% deviation range. BaNa performs slightly better than PEFAC for above 5 dB SNR, while PEFAC performs slightly better than BaNa for below 5 dB SNR. The GPE rate for BaNa with a 5% deviation

range is 30% at 0 dB, averaged over all 8 types of noise. The mean and standard deviation of Fine Pitch Errors (FPE) are also evaluated, using a 10% deviation range. The mean and standard deviation of FPE for BaNa are both 1.9% at 0 dB, which are only about 0.5% higher than the mean and standard deviation of FPE for PEFAC and HPS.

Since the SAFE algorithm is only trained to detect F_0 for speech with babble noise and white noise, we show its performance for these two types of noise at the end of this section, where we also present Wu's results, since it is unclear how to run Wu's code on long speech samples. Therefore, we only test the Wu algorithm for the LDC database. Since the -10 dB SNR and -5 dB SNR scenarios are very severe noisy environments, we present the rest of the F_0 detection results for noise conditions with SNR greater than or equal to 0 dB.

The GPE rates for speech with neutral emotion are shown in Figs. 2.6 and 2.7 for the CSTR and KEELE databases, respectively. Similar results are obtained for the proposed BaNa algorithm and the five other algorithms, with the main difference being that PEFAC at 0 dB SNR performs slightly better than BaNa for the CSTR database. With noise at 0 dB SNR, the GPE rate of BaNa is 35.4% for the CSTR database, and 20.3% for the KEELE database. However, since the ground truth F_0 values for the CSTR and KEELE databases are based on the laryngograph signals, we checked the ground truth values for a few speech samples and found that there are many spikes and discontinuities in the ground truth F_0 values found by using the laryngograph, especially on the boundaries of voiced and unvoiced frames. We can see from Figs. 2.6 and 2.7 that even at 20 dB SNR, the lowest GPE rate for all algorithms is still greater than 5%. While the ground truth for these databases may include several unvoiced frames and less reliable data, we present these results for the CSTR and KEELE databases in Figs. 2.6 and 2.7 in order to facilitate comparison with other F_0 detection algorithms that use these databases.

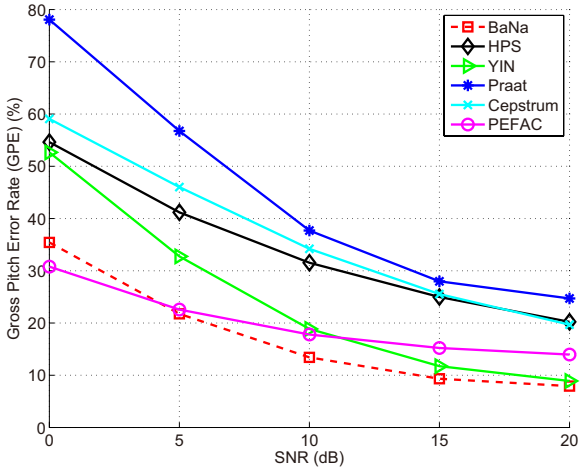


Figure 2.6: GPE rate of the different algorithms for the CSTR database [3], averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

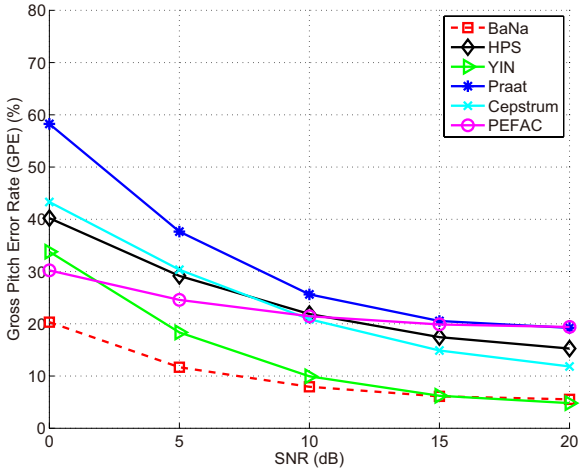


Figure 2.7: GPE rate of the different algorithms for the KEELE database [4], averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

Babble noise and white noise are the most common types of noise in speech processing. Since the SAFE algorithm is only trained on babble noise and white noise, we only compare the results of SAFE for these two types of noisy speech.

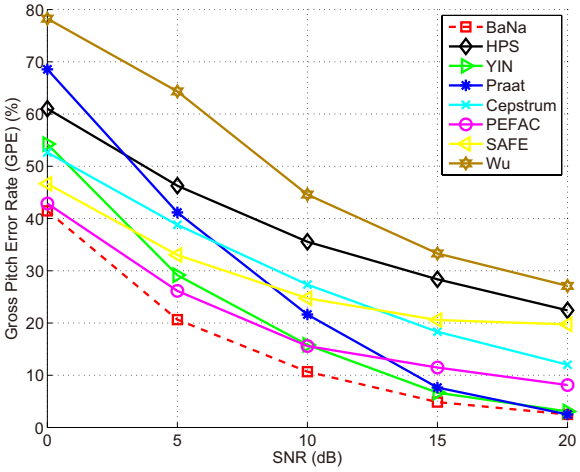


Figure 2.8: GPE rate of the different algorithms for the LDC database [1] for speech with babble noise. Detected F_0 deviating more than 10% from ground truth are errors.

The KEELE database is used for training of SAFE, as in [54], and the LDC database is used for testing. We also show the performance of the Wu algorithm proposed in [53]. The detected F_0 value is considered to be an error if it deviates more than 10% from the ground truth value, and again we use GPE rate as the error measurement metric. Figs. 2.8 and 2.9 present the GPE rate of the different algorithms for the LDC database for speech with babble noise and white noise, respectively. We can see that the F_0 detection for speech with babble noise is more difficult than F_0 detection for speech with white noise. Results show that BaNa, YIN, and PEFAC provide the lowest GPE rate for F_0 detection for speech with babble and white noise.

Speech with noise at 0 dB SNR is a challenging scenario for F_0 detection. For a head to head comparison, we present the performances of the BaNa algorithm and the closest competing algorithms, PEFAC and YIN, using the LDC database for eight different types of noise at 0 dB SNR in Fig. 2.10. We can see that BaNa has the lowest GPE rate for four out of eight types of noise. For the babble noise,

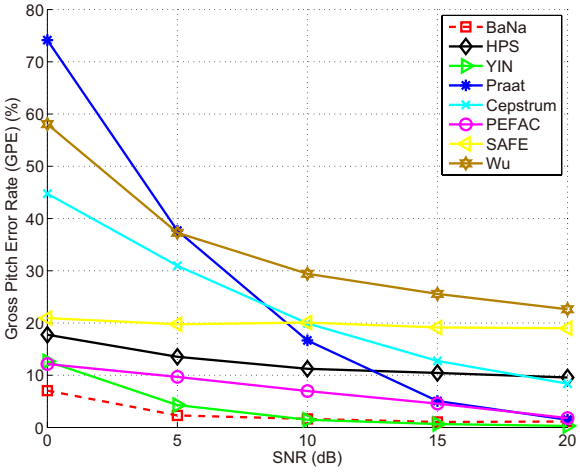


Figure 2.9: GPE rate of the different algorithms for the LDC database [1] for speech with white noise. Detected F_0 deviating more than 10% from ground truth are errors.

which is a very common type of noise in real life scenarios, the BaNa algorithm achieves a 41.5% GPE rate compared with PEFAC's 42.9% and YIN's 54.3%, even when the speech is only slightly audible by the human ear. We can also see from Fig. 2.10 that the babble noise and the destroyer operations noise cause the worst degradation in the F_0 detection performance. By investigating the spectrum of several noisy speech samples, we found that the high spectral peaks of these two types of noise concentrate in the same frequency range as the spectral peaks of speech. On the other hand, the high spectral peaks of high frequency noise, vehicle noise and white noise are distributed in the frequency range, which is quite different from the spectrum of human speech, making it easier to differentiate speech spectral peaks from noise spectral peaks. Therefore, the GPE rate for speech with these types of noise remains at a relatively low level even at 0 dB SNR.

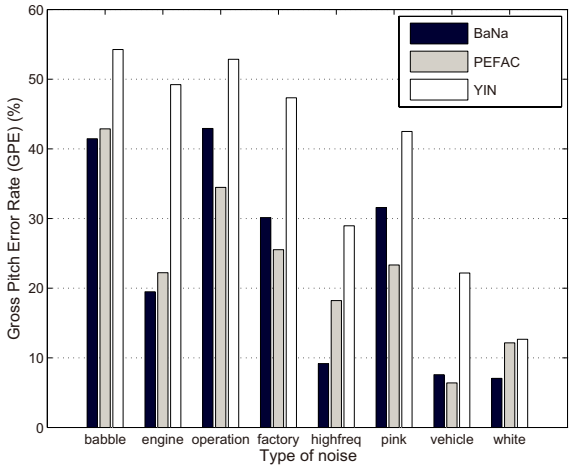


Figure 2.10: GPE rate of BaNa, PEFAC and YIN for the LDC database [1] with eight types of noise at 0 dB SNR. Detected F_0 deviating more than 10% from ground truth are errors.

2.5.2 Breakdown analysis of the BaNa algorithm

As we can see from the above F_0 detection performance for speech, the proposed BaNa algorithm has the most advantage at 0 dB SNR across almost all speech databases. To provide additional insights to understand the core design of this noise-resilient algorithm, as well as the differences between BaNa and other algorithms, we provide a breakdown analysis of BaNa here:

- BaNa only considers the frequency ratios among the lower-order harmonics, and also Cepstrum is included as one of the F_0 candidates, thus BaNa is less affected by octave errors than Schroeder's frequency histogram.

- Harmonic summation methods use the amplitudes of spectral peaks to weight the frequency histogram, which is not a noise-resilient approach, since noise peaks with high amplitudes are likely to be chosen as F_0 after the harmonic summation. The BaNa algorithm, on the other hand, only uses the peak amplitude information to choose the spectral peaks, but the F_0 candidates

calculation is solely based on the frequency ratios of the chosen peaks. No peak amplitude information is used at this point, as it may be severely corrupted by noise.

- By providing a tolerance range for these frequency ratios, our algorithm is able to combat the frequency drift of harmonics and shape distortions of harmonic peaks caused by the noise.

- Post-processing using the Viterbi algorithm in BaNa considers the F_0 continuity, which helps to choose the F_0 candidates more accurately.

- Since the F_0 candidates calculated from peak frequency ratios are only based on lower-order harmonics, adding the Cepstrum as an additional candidate helps to capture the general period information for all spectral peaks.

To show the effectiveness of using the Cepstrum candidate and the spectral peak with the lowest frequency as two additional F_0 candidates, and using the Viterbi post-processing, in Fig. 2.11 we plot the GPE rates for the BaNa algorithm, BaNa without the Cepstrum candidate, BaNa without the lowest frequency candidate, BaNa without both added candidates, and BaNa without post-processing for the LDC database. BaNa without post-processing means that we choose the F_0 candidate with the highest confidence score to be F_0 for each frame. We can see that using the two added candidates and using post-processing are effective to reduce the GPE rate. We can see that the GPE rate is as high as 20% when SNR is 20 dB without using both added candidates. This is because for some frames, only the F_0 peak's amplitude is high enough to be detected. Therefore, no F_0 candidates are derived from calculating frequency ratios.

By comparing the results for BaNa without post-processing with the results in Fig. 2.11 for the two algorithms that have no post-processing, HPS and Cepstrum, with the results in Fig. 2.5, we can see that BaNa without post-processing still

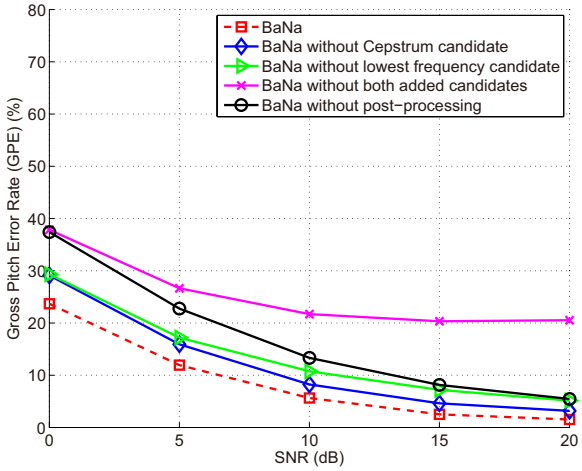


Figure 2.11: GPE rate of BaNa, BaNa without the Cepstrum candidate, BaNa without the lowest frequency candidate, BaNa without both added candidates, and BaNa without post-processing for the LDC database, averaged over all eight types of noise. Detected F_0 deviating more than 10% from ground truth are errors.

achieves a lower GPE rate. Thus, from the breakdown analysis we conclude that the post-processing is helpful, but it is not the most critical step in determining the performance of BaNa.

2.6 BaNa F_0 Detection Algorithm for Music

In this section, we extend the BaNa algorithm to enable F_0 detection of music signals in noisy environments.

2.6.1 Modifications on BaNa for F_0 detection for music

Since speech and music have different frequency characteristics, the BaNa algorithm needs to be slightly modified for F_0 detection in music. In Section 2.3.2, when detecting F_0 for speech, the p peaks with the lowest frequencies are selected. However, music signals can have high F_0 values, thus the low frequency region can

be dominated by noise peaks. Thus, if we still choose the p peaks with the lowest frequencies, noise peaks are chosen incorrectly. Therefore, for music F_0 detection, we select the p peaks with the highest amplitudes in the frequency range considered. We show the benefit of this change in Section 2.6.4.

2.6.2 Experimental settings for F_0 detection for music

Due to the variety of spectrum characteristics for different musical instruments, to show the performance of the F_0 detection algorithms for musical instruments, samples from four instruments are used: violin, trumpet, clarinet and piano. These music pieces are selected and downloaded from [80], which were all recorded in a quiet environment. These music pieces include a piece of 3.7 s long violin with 9 notes, a piece of 12.9 s long trumpet with 12 notes, a piece of 5.3 s long clarinet with 4 notes, and a piece of 7.8 s long piano with 8 notes. All the music samples used are also included in the BaNa toolkit [64]. The additive noise is from the same noise database as in Section 2.4.1.

For F_0 detection in music, we use hand-labeled ground truth F_0 values, which are determined by manually inspecting the spectrum and locating the F_0 peaks for each frame. Due to the large F_0 range in music, we use a more stringent F_0 deviation criteria for error measurement. The difference between two neighboring key frequencies is $2^{\frac{1}{12}}$, which is approximately 6%. Thus, we use half of this number, i.e., 3%, as the F_0 deviation criteria, which is also called the musical quarter tone [81]. Thus, detected F_0 values that deviate more than 3% from the ground truth values are counted as errors. This error measurement metric is also used by other studies [81].

2.6.3 Parameter tuning

According to the music F_0 range specified in [38], the lower and the upper limit for F_0 of music are set to $F_0^{min} = 50$ Hz and $F_0^{max} = 4{,}000$ Hz, respectively. It is set to 50-4,000 Hz for these competing algorithms as well for a fair comparison. The other parameters are the same as those in Table 2.2, and are not further optimized using music signals.

2.6.4 BaNa vs. BaNa music

To show the effectiveness of the changes made to the BaNa algorithm to be suitable for F_0 detection in music, we plot the GPE rate in Fig. 2.12 for a piece of violin music using both the original BaNa algorithm and the customized BaNa music algorithm with eight different types of noise at 0 dB SNR. The F_0 detection range is set to be the same for the original BaNa algorithm and the customized BaNa music algorithm, i.e., $F_0^{min} = 50$ Hz and $F_0^{max} = 4{,}000$ Hz. We can see that the modifications in the BaNa algorithm for music F_0 detection are necessary, and can greatly reduce the GPE rate for almost all types of noisy music. Note that throughout this section, we just use 'BaNa' to represent the BaNa music algorithm.

2.6.5 F_0 detection performance for music signals

In this set of experiments, we compare the BaNa algorithm with other algorithms for music F_0 detection. Within the evaluations of the SAFE algorithm in [54], there are no detection results for music. Therefore, we are not able to run the SAFE algorithm here due to the lack of noisy music training data. Also, according to the authors of PEFAC [50], PEFAC is not suitable for F_0 detection in music, hence we do not include that here. Also, it is unclear how to use the code for the

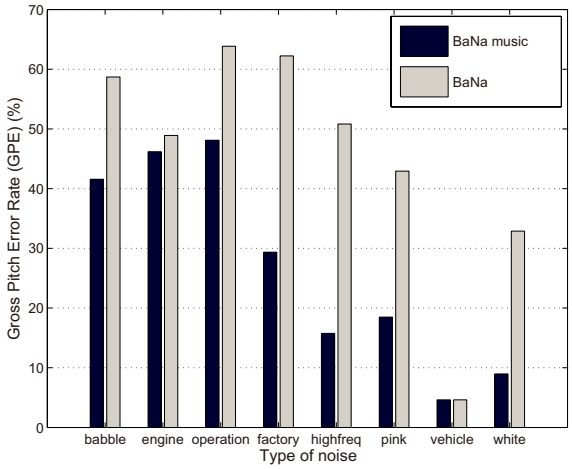


Figure 2.12: GPE rate of BaNa and BaNa music for a piece of violin music with eight types of noise at 0 dB SNR. Detected F_0 deviating more than 3% from ground truth are errors.

Wu algorithm [53] to process long audio samples. Therefore, we only compare the proposed BaNa algorithm with YIN, HPS, Praat, and Cepstrum. Figures 2.13-2.16 show the GPE rates of the different algorithms for violin, trumpet, clarinet, and piano, respectively, averaged over the eight types of noise. Results on all these four instruments show that the BaNa algorithm achieves the lowest GPE rate among all the algorithms. At 0 dB SNR, BaNa achieves the lowest GPE rates, which are 36.1%, 28.1%, 58.3%, and 35.3% lower than the closest performing algorithm, HPS, for violin, trumpet, clarinet, and piano, respectively.

From the above results, we can see that BaNa, HPS, and YIN provide the overall best F_0 detection performance in noisy music. Praat and Cepstrum do not provide consistent or satisfying results. Therefore, we choose BaNa, YIN, and HPS for detailed comparison using the violin piece with eight different types of noise at 0 dB SNR. In Fig. 2.17 we can see that BaNa has the lowest GPE rate for seven out of eight types of noise, especially for the speech babble noise.

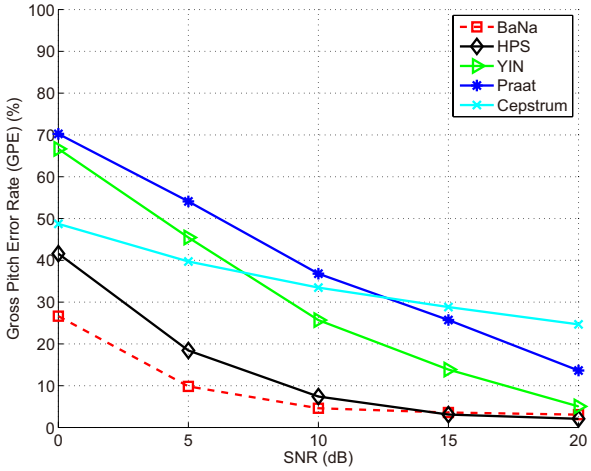


Figure 2.13: GPE rate of the different algorithms for a piece of violin music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

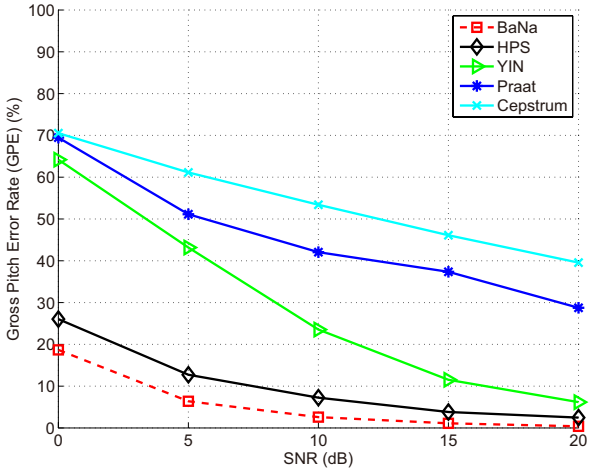


Figure 2.14: GPE rate of the different algorithms for a piece of trumpet music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

2.7 Implementation Issues

With an increasing number of speech-related smartphone apps emerging in the market, and due to the fact that speech captured by smartphones are usually

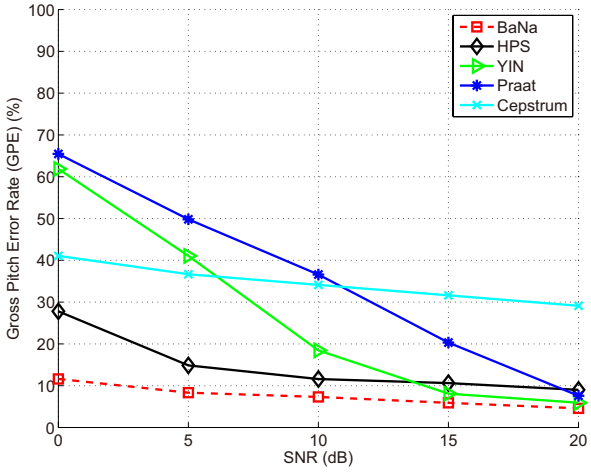


Figure 2.15: GPE rate of the different algorithms for a piece of clarinet music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

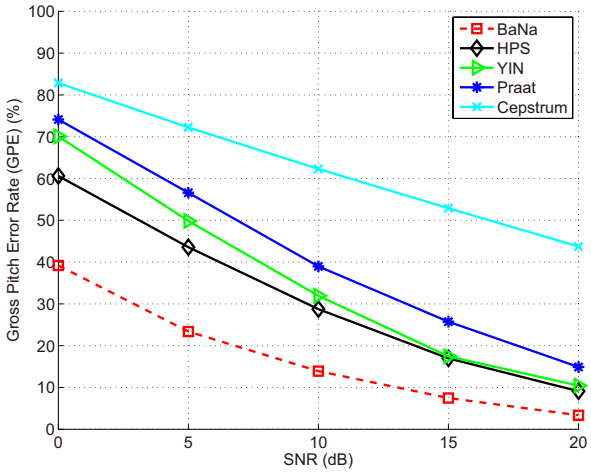


Figure 2.16: GPE rate of the different algorithms for a piece of piano music with eight types of noise. Detected F_0 deviating more than 3% from ground truth are errors.

affected by different types of noise, it is important to discuss the challenges in implementing the BaNa F_0 detection algorithm on a mobile platform. To explore these issues, we implemented BaNa as an app on an Android platform. The code

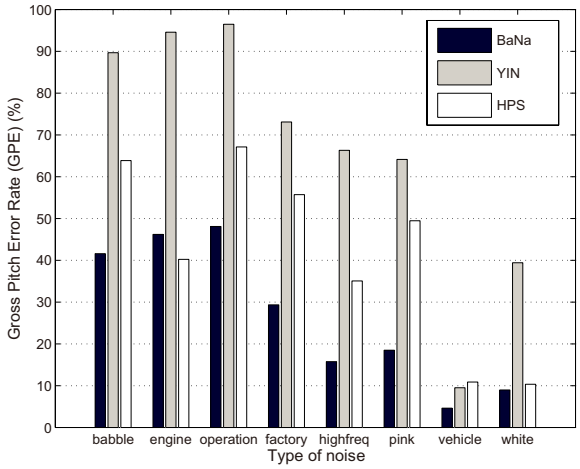


Figure 2.17: GPE rate of BaNa, YIN and HPS for a piece of violin music with eight types of noise at 0 dB SNR. Detected F_0 deviating more than 3% from ground truth are errors.

for the Android implementation of BaNa is available at [64]. The BaNa app is also available for free downloading on Google Playstore [82].

The user can either load a speech file that is stored on the phone, or record a short piece of speech. The screenshot on the left-hand side shows that the app detects F_0 for a 2.0 s long speech file with 8 kHz sampling rate in just 0.8 s. As shown on the right-hand side screenshot, some basic parameters in the BaNa algorithm can be adjusted by the user.

Since the F_0 candidates and their confidence scores can be calculated separately for each frame, as explained in Section 2.3.2, we can take advantage of multithreading to speed up the implementation. Single-core and multi-core devices can both benefit from multithreading through an increased utilization of the processor(s). When all threads finish the calculation of F_0 candidates for their own assigned frames, the Viterbi post-processing can go through all the frames to determine F_0 for each frame.

To test the speed of the BaNa F_0 detection implementation, we ran tests

Table 2.3: Elapsed time (in seconds) for F_0 detection using the BaNa algorithm implemented on an Android platform with a different number of threads and FFT sizes. The speech file is 1.3 s long.

Number of threads	FFT size			
	2^{16}	2^{15}	2^{14}	2^{13}
1	11.05	5.16	2.52	1.42
2	6.85	3.15	1.49	0.85
3	5.93	2.67	1.28	0.92
4	5.89	2.67	1.25	0.80

with different parameter settings and speech sample lengths on a Google Nexus 7. The specs of the device are: Nvidia Tegra 3 quad-core processor clocked at 1.2GHz, 1GB of RAM. Of course, the speed of the algorithm highly depends on the capabilities of the mobile device. Table 2.3 shows the elapsed time to process a 1.3 s long speech sample with sampling rate of 22,050 Hz. All the parameters for the BaNa algorithm are set to be the same as those in Table 2.2. For a more reliable measurement, the elapsed time for each test is averaged over 10 trials. We can see that the BaNa F_0 detection algorithm runs roughly 8 times faster by using the 2^{13} FFT size than using the 2^{16} FFT size, though using the 2^{13} FFT size still provides a reasonable frequency resolution of $22,050/2^{13} = 2.7$ Hz per sample. Also, we can see that multithreading helps to further reduce the elapsed time.

We show in Table 2.4 the elapsed time for F_0 detection for speech samples with different lengths. For this test, we choose the setting that provides the fastest speed, i.e., the number of threads is set to 4, and the FFT size is set to 2^{13}. These results show the possibility to turn the BaNa algorithm into a real-time F_0 detector even on mobile devices.

Table 2.4: Elapsed time (in seconds) for F_0 detection using the BaNa algorithm implemented on an Android platform for speech samples with different lengths.

Number of threads	FFT size	Length of speech sample (s)				
		2	4	6	8	10
4	2^{13}	0.91	1.61	2.39	3.05	3.82

2.8 Conclusions

In this chapter, we presented BaNa, a noise resilient hybrid F_0 detection algorithm for speech and music. BaNa was designed to detect F_0 in noisy environments, for example on a smartphone. This would enable the wide deployment of speech-based applications, such as the ones that use emotion detection. Evaluations show that BaNa achieves the lowest GPE rate for most cases among the algorithms investigated from the literature including YIN, HPS, Praat, Cepstrum, PEFAC, SAFE and Wu for different types of background noise, and under different SNR levels from -10 dB to 20 dB. Even for the very noisy scenario of 0 dB SNR, the GPE rate of BaNa averaged over all types of noise is only about 20% to 35% for speech for the different databases evaluated. The GPE rate for music at 0 dB SNR is 12% to 39% for different instrument pieces. Additionally, we implemented the BaNa algorithm on an Android platform, and implementation issues such as delay and multithreading are discussed. Tests on a real device show that the implementation is fast enough to provide for real-time F_0 detection applications.

In the next chapter, we use the proposed noise-resilient BaNa F_0 detection algorithm in a speech-based emotion classification system, in which speech samples' F_0 and other features are extracted for emotion classification. The emotion classification performance is evaluated on both clean speech samples and speech samples with different types of noise.

3 Enhanced Multiclass SVM with Thresholding Fusion for Speech-based Emotion Classification

3.1 Introduction

Emotions are a primary form of communication in humans and carry the potential to convey a wealth of information [83]. In particular, human speech contains rich information for effectively conveying emotions and communicating wants, needs, and desires. The richness of human speech for understanding emotions within human interactions has motivated researchers to explore the area of emotion classification based on speech [84].

Existing methodologies for assessing behavioral data for emotions are based largely upon using trained observational coders who manually decode different parameters in the speech signal according to some prescribed criteria [85]. This is very time intensive and requires hours of training as well as methods to ensure that coders are accurate and consistent with one another [86]. Furthermore, such procedures are costly from a time and financial standpoint and have the potential

to be subjective and error-prone. While prosodic features are easy to capture, and thus have been widely used in automatic emotion classification, mining useful emotion information solely from prosodic features is still a challenging task, and the classification accuracy is still not adequate [6] [5].

Therefore, improved emotion classification methods are needed, and a thorough analysis of the emotion classification accuracy under real scenarios is necessary, such as where modalities are captured in noisy environments. Speech has been used in conjunction with other modalities such as text [24] [87] [88], body gestures, and facial expressions to build multimodels for emotion classification [89] [90] [91], but in this chapter we focus on emotion classification based solely on vocal features.

There are a variety of applications that use speech-based emotion classification. Ticket reservation systems employ emotion detection to recognize annoyed or frustrated customers and respond accordingly [92]. Call centers employ emotion classification to prioritize impatient customers [93] [94]. Warning systems have been developed to detect aggressive driving [32] or to keep the driver alert [95]. In the healthcare field, emotion classification is used by clinicians for assessment or treatment of patients with psychological disorders or conditions that create emotional difficulties, such as autism or depression [96] [97]. Speech-based emotion sensing technologies have been implemented on mobile devices, such as smartphones, for behavioral studies [5] [33] or patient monitoring [34]. Emotion attribute can also be used for speaker recognition [98] or emotional speech synthesis [99] [100] [101] [102].

The emotion classification system used in this work extracts the speech signal's fundamental frequency, energy and other speech features, and the widely employed Support Vector Machine (SVM) learner is used for One-Against-All (OAA) classification for each emotion. To improve the classification performance, we use the thresholding fusion mechanism proposed in [103], which fuses confidence scores

from multiple OAA classifiers by comparing the highest confidence score with a pre-set threshold to determine whether to classify the sample or reject it.

Initial results using this system were presented in our previous work [17], which, however, we subsequently found contained erroneous results due to an issue with the voice feature data that was used in the classification. In this work, we have corrected the problem and also changed the SVM kernel function to be Radial Basis Function (RBF), instead of the hybrid kernel proposed in [17]. We added Mel-Frequency Cepstral Coefficients (MFCCs) and speaking rate to the speech feature set. Additionally, feature selection and over-sampled methods were used to further improve the classification performance. More thorough evaluations and discussions are presented in this work as well.

Our method achieves a decision-level correct classification rate of 80% for six emotions using the LDC dataset [1] spoken by actors and actresses, and 45% on a noisy dataset spoken by ordinary speakers using the UGA dataset [64]. Our system outperforms a state-of-the-art method proposed in [5], which achieves a decision-level correct classification rate of 71% for classifying five emotions based on the same LDC dataset. Our system allows defining a confidence threshold level to improve the performance at the expense of rejecting more samples as unclassified. As an example, the decision-level correct classification rate can be increased to 93% and 56% when half of the samples are rejected as unclassified for the aforementioned LDC dataset and UGA dataset, respectively. The MATLAB code for our emotion classification system is available on the University of Rochester Wireless Communications and Networking Group's website [64].

The contributions of this work are:

- We build upon our preliminary work [17] to construct a complete and effective speech-based emotion classification system, by employing more features (e.g., MFCC and speaking rate) and adding three performance enhancement strategies (i.e., speaker normalization, training using over-sampled datasets,

and feature selection). We also conduct a thorough comparison with state-of-the-art methods and a systematic analysis of system components in different scenarios (e.g., general tests and gender dependent tests).

- We employ the thresholding fusion mechanism proposed in [103] to further improve the emotion classification accuracy at the expense of rejecting some uncertain speech samples. We illustrate that this strategy will be beneficial in many practical situations.

- We investigate the emotion classification performance on real scenarios including speaker-independent tests, tests on noisy speech signals, and tests using a dataset with non-professional acted emotions.

The rest of this chapter is organized as follows. Section 7.3 provides a brief survey of two components of a speech-based emotion classification system, i.e., speech features and classifiers. Section 3.3 describes our proposed emotion classification system, including the thresholding fusion method and three performance enhancement strategies. Section 3.4 explains the speech datasets and evaluation metrics used in this work. Extensive experimental results of the system using different databases and different scenarios are presented in Section 7.6. Finally, Section 7.7 concludes this chapter.

3.2 Related Work

A speech-based multiclass classification system consists of two components: a set of speech features to extract from the speech signals of the dataset and a classifier to classify the speech signals based on their extracted features. Therefore, we survey existing emotion classification techniques according to these two aspects.

3.2.1 Speech features

An important issue in the design of a speech-based emotion classification system is the extraction of suitable features that efficiently characterize different emotions and perform consistently, regardless of the speaker.

For speech analysis applications, such as emotion classification, speech recognition, and speaker recognition, a number of speech features have been commonly used. In the time domain, popular prosodic features are energy, speaking rate, duration, and zero crossing rate. In the frequency domain, spectral features represent vocal cord and vocal tract system characteristics. For example, the authors of [104] found that emotions with high arousal, such as anger and happiness, result in higher mean values of the first formant frequency in all vowels, whereas emotions with positive valence, such as happiness and pride, result in higher mean values for the second formant frequency. Some spectral features, such as Mel-Frequency Cepstral Coefficients (MFCCs) and Perceptual Linear Predictive (PLP) coefficients, are derived on the concept of logarithmically spaced filter banks matched to the human auditory system. These features work well for the nonlinearities of speech signals, which are caused by the turbulence during speech production for example, that cannot be accurately described by linear source-filter speech models. Additionally, fundamental frequency (F_0) and energy are closely related to emotion classification. Some other commonly used spectral features include energy slope, and Log Frequency Power Coefficients (LFPC). The difference, delta, and acceleration values of these features are also used to capture the temporal dynamics of the speech signals.

For speech-based emotion classification studies, the work proposed in [5] uses Perceptual Linear Predicative (PLP) coefficients as speech features. Speech features F_0, intensity, first formant frequency, voice quality measures, and MFCCs are used in [6]. A new speech feature called weighted frequency is proposed in [7],

which is representative of the spectral region containing the most energy. Besides weighted frequency, the speech features used in [7] include zero crossing rate, F_0, and energy. Long-term spectro-temporal features are used for emotion classification in [105]. Some psychology and behavior studies also adopt speech features such as F_0, energy, and speaking rate [106] [107] [108].

3.2.2 Emotion classifiers

For multiclass emotion classification systems, commonly used generative classifiers include Naive Bayes and Gaussian Mixture Models (GMM), for which the feature distributions for each emotional state are modeled. Commonly used discriminative classifiers, which do not employ any probability density modeling, include Support Vector Machines (SVM), k-Nearest Neighbors (kNNs), Multi-layer Perceptron (MLP), and decision tree. Sequential classifiers, such as Hidden Markov Model (HMM) based classifiers, have been used as well due to the advantage of reflecting the temporal dynamics of the speech features by using the state transition probability.

Early studies such as [109] implemented HMM for emotion classification, while in [5] [7] [110], GMM was implemented for the same purpose. An extended version of GMM for emotion classification was proposed by H. Tang et al. [111] by introducing a boosting algorithm for a reliable and accurate estimation of the class-conditional GMM. SVM has been recently widely used in speech-based emotion classification studies as well [28] [6] [112] [113].

3.3 Emotion Classification System

In this section, we present our multiclass SVM system for speech-based emotion classification. In order to improve the classification performance, we use three

enhancement strategies: speaker normalization, feature selection, and using over-sampled datasets for OAA SVM training. The effectiveness of using these strategies is investigated in Section 3.5.2. A thresholding fusion mechanism is also used, which provides the functionality to effectively increase the classification accuracy at the expense of rejecting some samples as unclassified.

3.3.1 Speech features

We divide each speech utterance into 60 ms segments with 10 ms time shifts, and only extract speech features for the voiced segments. The following describes what features are used and how they are extracted:

- **Fundamental frequency ($\mathbf{F_0}$)**: we use the noise-resilient BaNa F_0 detection algorithm [15] [16] to extract the F_0 values.
- **Energy**: we calculate the energy for each segment by taking the summation of all the squared values of the samples' amplitudes.
- **Difference of $\mathbf{F_0}$** and **difference of energy**: the difference of F_0 or energy values between two neighboring segments. More fluctuations may indicate active emotions, such as happiness or anger.
- **Frequency and bandwidth for the first four formants**: we use the linear predictive coding method for formant calculation.
- **Mel-frequency Cepstral Coefficients (MFCCs)**: we use the VOICEBOX toolkit [114] to find the 12 MFCCs for each speech frame.
- **Speaking rate**: measured in the number of syllables per second. We use the method described in [115].

Since speaking rate is measured on each speech utterance, and the other features are measured on each 60-ms frame, we calculate five statistics: the mean,

maximum, minimum, range, and standard deviation for each feature vector except speaking rate, resulting in $24 \times 5 + 1 = 121$ attributes that are sent to the classifier.

3.3.2 Speaker normalization

The characteristics of speech features differ from person to person, which increases the difficulties of speech-based emotion classification and speech recognition related research. For example, speaking with a higher tone, i.e., a higher F_0, is often a sign of active emotions, such as happy or anger. However, some speakers' average F_0 is higher than others'.

As we intend to analyze emotion independent of the speaker, speaker normalization is used as an enhancement strategy to reduce inter-speaker variability and increase the classification accuracy [116]. Speaker normalization aims to narrow the difference in speech features between speakers, and only retain the differences between emotion categories. Speaker normalization was first introduced by L. Lee and R. Rose [117] for frequency warping procedures. Later on, speaker normalization showed its benefits in the areas of both automatic speech recognition [118] and speech-based emotion classification [119].

The z-score normalization method [120] has been widely used to eliminate the difference between speakers. We calculate the mean and standard deviation of a specific feature across all frames of the utterances across all the emotions for each speaker individually. Then, each feature value is z-score normalized using the mean and standard deviation.

3.3.3 Feature selection using mutual information

Feature selection techniques have been used in emotion classification problems to reduce irrelevant or highly correlated features [121] [122] [123]. Mutual informa-

tion, as one of the techniques which is calculated between each feature and the class label, has been widely used since mutual information measures arbitrary dependencies between random variables, which makes it suitable for assessing the "information content" of features in complex classification tasks [124]. Therefore, mutual information is used on the speaker-normalized feature values to select the the most relevant features as well as to prevent the learner from overfitting. The MATLAB implementation of mutual information that we use here is from [125].

3.3.4 OAA SVM multiclass emotion classification

We choose SVM as our classifier. Compared to generative models such as GMM, SVM has been shown to have better discrimination power [28]. Compared to other discriminative models such as linear regression, SVM can use kernel functions to deal with linearly inseparable data [126].

Two approaches are commonly used to construct a multi-class SVM classifier by combining results from a number of ordinary binary SVMs: One Against All (OAA), which constructs one SVM per class to distinguish it from all the other classes, and One Against One (OAO), which constructs one SVM to distinguish each pair of classes. In this study, we choose the OAA approach for the sake of a better classification accuracy. There are two main approaches to combine these binary decisions from multiple OAA classification models, i.e., *and*ing binary decisions [127], or choosing the class with the largest confidence value [103]. We use the latter approach in this work to make sure that only one emotion is classified.

We use the default C parameter of the box constraint for the soft margin in the SVM. The Radial Basis Function (RBF) kernel is used, and the scaling factor σ in the RBF kernel is optimized to be 5. Sequential minimal optimization (SMO) [128] is used for a faster training of SVM.

3.3.5 Over-sampled training set

Another performance enhancement strategy employed in our emotion classification system is to use over-sampled datasets for OAA classifier training using the SMOTE method [129]. For multiclass classification problems, there are three approaches to train each individual OAA SVM classifier, i.e., using the raw uneven datasets, over-sampling the minority class, and under-sampling the majority class.

Take the 'happy or not' OAA classifier as an example. To train the classifier using the raw uneven dataset, all samples with happiness emotion are used as positive samples, and all samples from the other five emotions are used as negative samples. Studies show that the OAA class-boundary learned by imbalanced datasets can be severely skewed towards the positive class. As a result, the false-negative rate can be excessively high [130].

To train the classifier using an over-sampled dataset using SMOTE [129], $m-1$ synthetic samples are generated around each minority class training sample in the feature space, where m denotes the number of emotion classes. Since for the LDC dataset [1] we use in this work, the numbers of samples for different emotion classes are approximately the same, the numbers of samples of the majority class and the minority class are approximately the same after over-sampling, resulting in a balanced training set for the OAA SVM classifier.

To train the classifier using an under-sampled dataset, all samples with happiness emotion are used as positive samples, and the same number of randomly selected samples from the other five emotions are used as negative samples.

3.3.6 Thresholding fusion mechanism

We extend our emotion classification system by adding a thresholding fusion mechanism module [103]. When the module is off, we always classify a sample to the class with the highest confidence value, as described in Section 3.3.4. When the

module is on, however, we output the class label only when the highest confidence exceeds a certain threshold; otherwise, we reject classifying this sample.

This thresholding fusion mechanism essentially avoids classifying difficult samples in order to achieve high accuracy in the samples that are classified. This will make the system more robust in practice. As the system classifies emotion at the utterance level, oftentimes it is better to classify fewer utterances with a higher accuracy than classifying all utterances with many classification errors. Take the aggressive driver detection system as an example. The system should take interventions or issue warnings only if it is very confident that the driver is in a very emotional state. Similarly, in a behavioral study, it is more important to have reliable estimates of the participants' emotional states during a few times rather than continuous estimation that is less accurate. Therefore, utilizing a rejection strategy is beneficial for many emotion classification systems.

Figure 3.1 illustrates our emotion classification system. In the learning phase, for each utterance, the extracted speech features and the emotion labels are used to train each individual OAA SVM model X_i, where i = *1,2,...,m*, and m denotes the number of emotion classes. In the testing phase, speech features of the testing utterance j are extracted and then sent to each trained model X_i, resulting in confidence value $C_{X_i}(j)$, where j = *1,2,...,n*, and n denotes the number of testing utterances. Assuming that model X_p yields the highest confidence measure for utterance j, the confidence measure $C_{X_p}(j)$ is then compared against a user-controlled confidence threshold γ to decide whether to reject the sample as unclassified. We show in Section 3.5.3 that setting the confidence threshold γ to a higher value can result in a higher emotion classification accuracy. However, more instances are left as unclassified.

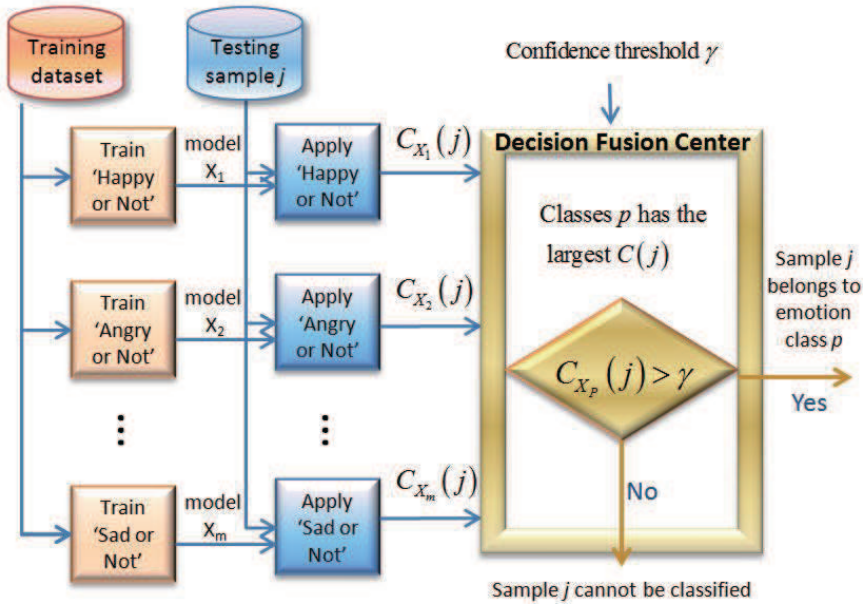


Figure 3.1: Our emotion classification approach using OAA SVM with thresholding fusion.

3.4 Datasets and Evaluation Metrics

Before we present the experimental evaluation of our emotion classification system, we explain the speech datasets and the evaluation metrics used in this work.

3.4.1 LDC and UGA datasets

To train our emotion classification system as well as to test its performance, we select two speech emotion analysis datasets, which are the LDC dataset [1] and the UGA dataset [64]. A few samples of the speech utterances from the LDC and the UGA datasets are available on the University of Rochester Wireless Communications and Networking Group's website [64], to provide readers with a better understanding of these two datasets.

For English speech-based emotion analysis that our system is designed for, the LDC dataset [1] is one of the standard benchmark datasets [6] [5]. The advantage

of using this library is that the emotions generated by professionals are expressed more explicitly compared to speech recorded by ordinary people. An alternative is to use speech material from movies [131] or recordings of everyday life. However, it is difficult to determine appropriate reference labels, since many natural utterances are emotionally ambiguous.

The LDC dataset includes a collection of speech files recorded by professional actors and actresses reading semantically neutral-meaning utterances such as dates and numbers spanning fourteen distinct emotion categories. Each utterance is between one and two seconds in length. Six emotions are selected in our emotion classification study as in [6]: disgust, happiness, sadness, anger, fear and neutral. There are three male speakers and four female speakers in the LDC database. About 15-25 utterances are spoken by each speaker for every emotion category, and there are 727 utterances in the LDC dataset in total.

The UGA dataset contains utterances spoken by students from the University of Georgia. Similar with the LDC dataset, the utterances in the UGA dataset are also dates and numbers. The same six emotions are acted by each one of the 133 students, and 10,489 utterances are included in the UGA dataset in total. Though more data can be used to train the emotion classification system, the diverse ways of expressing emotions by different speakers raises a challenge to the system as well. Also, people who are not actors or actresses tend to convey their emotions in a more implicit way, which makes it more difficult to classify emotions based only on speech. Additionally, the data is much noisier in the UGA dataset than it is in the LDC dataset.

3.4.2 Evaluation metrics

Since different state-of-the-art emotion classification systems use different performance evaluation metrics, to compare our system with these systems, we explain

several evaluation metrics as follows.

We define the ratio of unclassified instances in the test set as *rejection rate*. We can vary the rejection rate by tuning the threshold parameter γ from Fig. 3.1.

To measure the average classification performance for all classified emotions after fusion, we define the metric 'decision-level (DL) correct classification rate' as:

$$DL\text{-}\%correct = \frac{\sum_{i=1}^{m} Dtp_i}{N}, \tag{3.1}$$

where Dtp_i denotes the number of decision-level true positive utterances for emotion i, m denotes the number of emotion classes, and N denotes the total number of utterances.

To evaluate the emotion classification performance for each individual emotion, we use the metric 'decision-level (DL) recall for emotion i', which is defined as:

$$DL\text{-}recall_{Emotion_i} = \frac{Dtp_i}{Dtp_i + Dfn_i}, \tag{3.2}$$

where, as defined in (3.1), Dtp_i and Dfn_i denote the number of decision-level true positive and false negative utterances, respectively, for emotion i.

The classifier-level (CL) accuracy is used to evaluate the performance for each individual OAA classifiers. Note that this value is not used for the final emotion classification, which is derived after fusing the OAA binary decisions. The 'CL-accuracy' for classifier X_i, i.e., for the classification of an instance as 'Emotion i or Not' is defined as:

$$CL\text{-}accuracy_{X_i} = \frac{Ctp_i + Ctn_i}{N}, \tag{3.3}$$

where Ctp_i and Ctn_i denote the number of classifier-level true positive and true negative utterances for emotion i, respectively. N denotes the total number of utterances.

3.5 Emotion Classification Performance

In order to analyze the performance of our OAA SVM-based thresholding fusion emotion classification system, we first compare the proposed full system with three state-of-the-art studies when no data is rejected. Then we show the effectiveness of using the three enhancement strategies through evaluations. Additionally, the performance improvement by using the thresholding fusion mechanism is presented for a general test and gender-dependent tests. Finally, the performance on more challenging scenarios is evaluated, including a speaker-independent test, a test on noisy speech samples, and a test on speech samples from ordinary speakers. Unless noted otherwise, for each individual testing scenario, we present our results using 80 selected features, a small feature set that still provides a relatively high emotion classification accuracy.

3.5.1 Comparison with state-of-the-art systems

We first compare the performance of our emotion classification system with three state-of-the-art emotion classification methods that were also evaluated using the LDC dataset. The LDC dataset is used for both the training and testing through seven rounds of cross-validations. A summary of the comparison of the systems is presented in Table 3.1. Unlike SVM, the GMM classifier used in [5] and [7] is not a binary classifier. Therefore, no over-sampling is needed, and thus we leave the entries for these two reference system as N/A in Table 3.1.

We provide either decision-level emotion classification recall or classifier-level emotion classification accuracy depending on what metrics were provided by the reference systems.

Table 3.1: Comparison of our system and several state-of-the-art emotion classification systems that also use the LDC dataset.

System	Dataset	Features	Classifier	Speaker norm.	Over-sampling	Feature selection	Frame length
Our system	LDC	F_0, F_0 difference, energy, energy difference, frequencies and bandwidths for F_1-F_4, speaking rate, MFCCs	SVM	Yes	Yes	Yes	60 ms
Rachuri et al. [5]	LDC	32 perceptual linear predictive coefficients (static and delta values)	GMM	No	N/A	No	30 ms
Bitouk et al. [6]	LDC	F_0, F_0 delta, F_1, energy, energy delta, jitter, shimmer, relative spectral energy above 500 Hz, duration of voiced segments, MFCCs and duration over different phoneme regions	SVM	Yes	No	Yes	25 ms
Sethu et al. [7]	LDC	zero crossing rate, energy, F_0, weighted frequency	GMM	Yes	N/A	No	40 ms

Performance comparison for general test

The work in [5] classifies five emotions: anger, sadness, neutral, happiness and fear. In [5], similar narrow emotions in the LDC dataset are clustered to the above five broad emotion categories. For example, three narrow emotions, elation, happiness, and interest, are grouped into a broad happiness emotion category. However, in our study, only samples in the narrow happiness category are used for happiness. Decision-level recall values, as defined in (3.2), are used to evaluate the emotion classification performance for each emotion.

Figure 3.2 compares the decision-level recall for each individual emotion for our system with that obtained by the method in [5]. Since disgust is not among the emotion categories evaluated in [5], we leave the result for disgust for the reference system blank in the figure. We can see that our system outperforms the method in [5] for four emotions, i.e., anger, sadness, happiness, and fear.

Another method, proposed in [6], also classifies the same six emotions as our

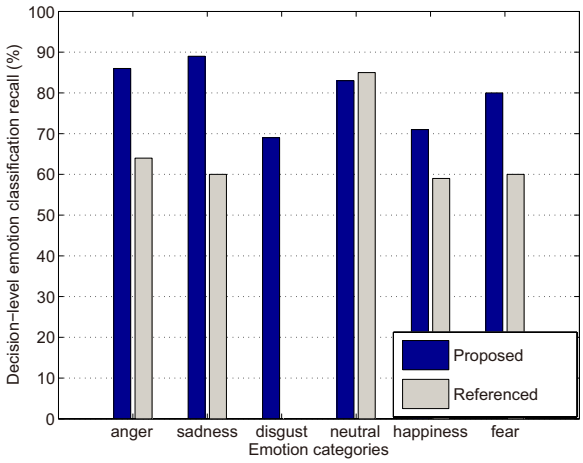


Figure 3.2: Decision-level emotion classification recall (%) for each individual emotion for our system without rejecting any samples and the method in [5], using the LDC dataset. Speaker normalization, feature selection, and over-sampled training sets are used.

work using the LDC dataset. They use classifier-level (CL) accuracy, as defined in (3.3), to evaluate the performance for each OAA classifier in their system. Hence, in Fig. 3.3, we compare the classifier-level emotion classification accuracy for our system with that obtained by the method in [6] for each individual emotion. From [6], we use the results derived by using their best setting, i.e., 'combined features', which is class-level spectral features plus utterance-level prosodic features. We can see in Fig. 3.3 that the classifier-level accuracy for our system outperforms the results in [6] for all individual OAA classifiers.

Performance comparison for speaker independent test

The authors in [7] classify five emotions: anger, sadness, neutral, happiness, and boredom using the LDC dataset. In Fig. 3.4, we compare the decision-level recall for each individual emotion for our system with that obtained by the method in [7]. For [7], we use the results obtained by using their best feature set 'ZEP+WF'

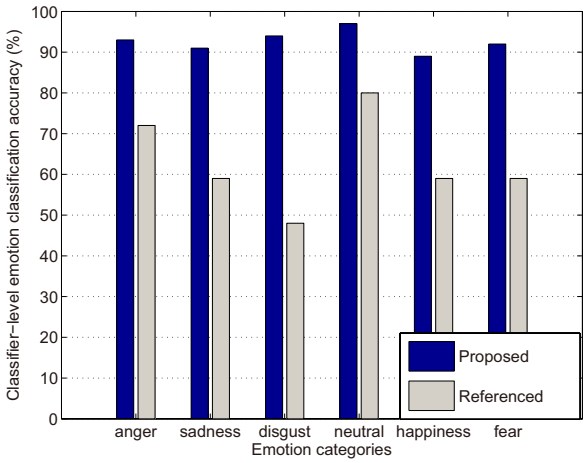


Figure 3.3: Classifier-level emotion classification accuracy (%) for each individual emotion for our system without rejecting any samples and the method in [6], using the LDC dataset. Speaker normalization, feature selection, and over-sampled training sets are used.

(including zero crossing rate, energy, pitch, and weighted frequency). Since disgust and fear are not among the emotion categories evaluated in [7], we leave the results for disgust and fear for the reference system blank in the figure. Comparing the results for the speaker-independent test in Fig. 3.4 with the results for the general test in Fig. 3.2, we can see that our system performance drops greatly when no data from the user has been used for training. Compared with the results derived by [7], our system provides higher decision-level recall values for sadness, neutral, and happiness, but lower decision-level recall values for anger.

3.5.2 The effectiveness of three enhancement strategies

Our proposed system contains three performance enhancement strategies, i.e., speaker normalization, feature selection, and over-sampling the training set. In order to gain a better understanding of the system, it is important to analyze the effectiveness of each individual strategy on the final system performance. In this

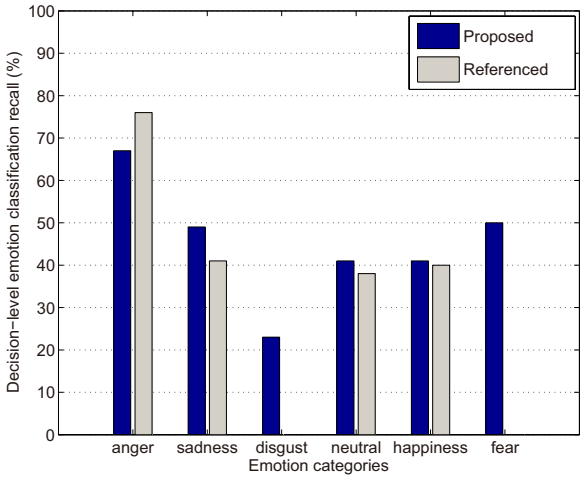


Figure 3.4: Decision-level emotion classification recall (%) for each individual emotion for our system without rejecting any samples and the method in [7] for the speaker-independent test using the LDC dataset. Speaker normalization, feature selection, and over-sampled training sets are used.

section, we evaluate the effectiveness of these three strategies. The LDC dataset is used for both the training and testing through seven rounds of cross-validations.

First, we compare the decision-level correct classification rates, as defined in (3.1) with and without speaker normalization when no data is rejected. All features are used in this evaluation, and the training set is over-sampled using SMOTE [129], as described in detail in Section 3.3.5. Due to the randomness in the synthetic sample generation process in the SMOTE algorithm, we generate the over-sampled training set for five different trials, and the performance is calculated by averaging these five trials. Results show that using speaker normalization achieves a decision-level correct classification rate of 81.0%, which is slightly higher than the result without using speaker normalization, which is 80.5%.

Second, we evaluate the benefit of using feature selection. We compare the emotion classification performance using the LDC dataset with a feature set chosen by mutual information and a referenced feature set with randomly selected

features, respectively. Features are randomly selected five times for the referenced feature set, and the average results are calculated. An over-sampled dataset is used for training, as explained in Section 3.3.5, and speaker normalization is used.

Results show that using features selected by mutual information achieves about 5 percentage points higher classification rate than using randomly selected features when 20 features are selected. This difference becomes smaller as the number of selected features decreases. However, using features selected by mutual information sometimes can achieve a lower classification rate than using randomly selected features. This is because the features are selected independently from each other, and features are selected only based on their mutual information to the class label. Thus "the m best features are not the best m features" [132]. Therefore, using mutual information cannot guarantee that the optimal feature set, which provides the highest correct classification rate, is selected.

We calculate the correct classification rates using different numbers of selected features using the LDC dataset, when all samples are classified. We find that using 80 out of 121 selected features can already provide a relatively high emotion classification rate, while one third of the features are not used, which reduces the computational complexity of the SVM classification.

To investigate which features are the most relevant to emotion classification, we illustrate in Table 3.2 which features are selected as the total number of features changes. We can see that MFCCs account for the largest portion of the selected features. Additionally, almost all energy and F_0 features are included in the selected feature set. Formants and speaking rate features are the last ones included in the feature set as the total number of selected features increases.

Finally, we compare the correct classification rates when using the three sampling methods for generating the training set as presented in Section 3.3.5, using all the features and speaker normalization. Results show that using an over-sampled dataset for OAA classifier training achieves a slightly higher correct classifica-

Table 3.2: The number of selected features for different sizes of the feature set for the general test using the LDC dataset. The total number of features is 121. Speaker normalization and over-sampled training sets are used.

Number of selected features	20	40	60	80	100	121
F_0 and difference of F_0	7	8	9	9	9	10
Energy and difference of energy	6	7	8	8	8	10
Formants	0	1	6	18	33	40
MFCCs	7	24	37	45	49	60
Speaking rate	0	0	0	0	1	1

tion rate of 81.0%, than using the raw uneven dataset, which achieves a correct classification rate of 79.6%. Due to the reduced number of samples in the under-sampled training set, the under-sampling method achieves the lowest classification accuracy of 75.1%.

3.5.3 Performance evaluation using thresholding fusion

In order to determine how much benefit is gained by using thresholding fusion and thereby leaving some samples as unclassified, we performed several tests. First, we evaluate the performance as we increase the threshold and thus increase the number of rejected samples using the entire LDC dataset. Then, we examine the impact of the thresholding fusion using gender-dependent tests, where only the female (or male) samples from the LDC dataset are used for training and testing.

General test

The LDC dataset is used for both the training and testing through seven rounds of cross-validations. Figure 3.5 shows the decision-level correct classification rate when we change the rejection rate by tuning the confidence score threshold γ.

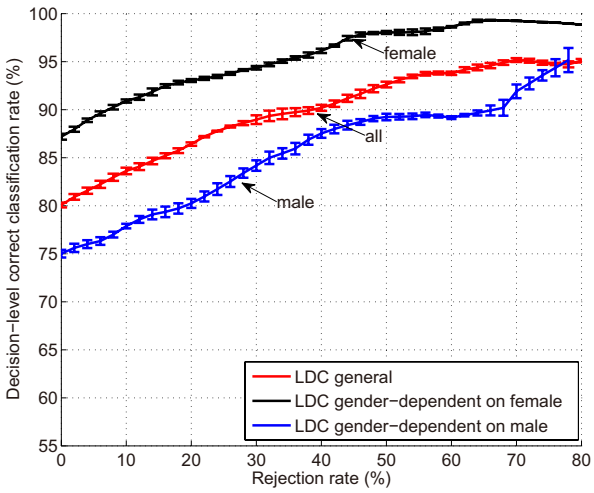


Figure 3.5: Decision-level correct classification rate vs. rejection rate for the general test and the gender-dependent tests using the LDC dataset with speaker normalization, feature selection, and over-sampled training sets.

80 features are selected for this test. When most of the testing data is rejected, very few samples are left, and we cannot obtain a reliable classification performance. Thus, we do not show the decision-level correct classification rate when the rejection rate is above 80%.

As we can see from Fig. 3.5, the decision-level correct classification rate generally increases as a higher confidence threshold is used, and hence, as more data is rejected. This number can be increased to 93% when 50% of the data is rejected. Therefore, using the thresholding fusion method can provide a more reliable emotion classification at the expense of leaving some data unclassified. As discussed previously, this can be valuable for a number of applications that provide actions based on the classification outcome. For these applications, it is much more important that the classification is accurate than it is to classify every sample with a lower accuracy.

Gender-dependent tests

Speech features differ between male and female speakers. For example, the F_0 of speech varies from 40 Hz for low-pitched male voices to 600 Hz for children or high-pitched female voices [70]. In order to illustrate how gender affects the emotion classification performance, we compare the results for gender-dependent tests with those for the previous general test that uses both male and female samples for both training and testing. For the gender-dependent tests, cross-validation is performed on all the samples for one gender for training and testing.

Figure 3.5 also shows the decision-level correct classification rate with different rejection rates for the gender-dependent tests on male and female speakers, respectively. Note that the features are selected only based on male or female speech utterances, and thus the top selected features are not the same.

The gender-dependent emotion classification performance for females is higher than that for males. Since the number of samples for female speakers is larger than that for male speakers in the LDC dataset, we have also tried using the same number of samples for both genders to train the model. Results are not shown in this chapter, but similar conclusions are obtained that the gender-dependent test for females provides better results than those for males. Another important result that we can see from this data is that the thresholding fusion mechanism improves the performance for males significantly, which is important since the performance for males when the thresholding fusion module is off is much lower than for females.

3.5.4 Performance evaluation for more challenging scenarios

In this section, we analyze the performance of our system in more challenging scenarios, namely when the speaker is not included in the training set (speaker-

independent test), when the speech data is noisy, and using data from the UGA dataset with non-professional acted emotions.

Speaker-independent test

Emotions are expressed in different ways by different speakers, and the speech features for different speakers vary as well. Thus, to get an idea of how our system performs when it is used on a new speaker, we run a speaker-independent test, where data from the tested speaker is not used in the training phase.

For the speaker-independent tests, we use the same 80 features as the feature set used for the general test. Figure 3.6 shows the decision-level correct classification rate for the speaker-independent tests using the LDC dataset with the over-sampled training dataset and speaker normalization. The legend denotes the initials of the seven speakers, where the gender of the speakers is added after the initials as 'm' for male speakers and 'f' for female speakers. We also show the speaker-independent result averaged over all seven speakers.

As shown in Fig. 3.6, the decision-level correct classification rate increases from 47% when no data is rejected to 65% when 80% of the data is rejected. Compared with the general test results shown in Fig. 3.5, the decision-level correct classification rate drops by about 33 percentage points when no samples from the target speaker are included in the training set. This shows the need for prior training with the subjects to achieve a good performance in our emotion classification system.

Test on noisy data

For emotion classification in real scenarios, noise is a factor that inevitably needs to be considered when we evaluate the system performance. We add babble and white noise to the LDC speech signals to generate a noisy dataset. The noise

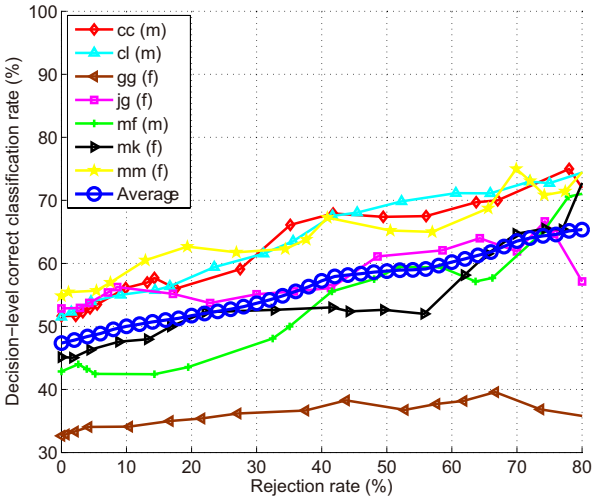


Figure 3.6: Decision-level correct classification rate vs. rejection rate for the speaker-independent test using the LDC dataset. Speaker normalization, feature selection, and over-sampled training sets are used.

database we use is [65]. A moderate noise level, i.e., noisy data at 5 dB Signal-to-Noise Ratio (SNR), is used for testing.

In order to classify emotions on noisy data, there are two approaches to train the system: using clean data or noisy data. In Fig. 3.7, we compare the results for both approaches with the results for training and testing on clean data. Results are shown for training on an over-sampled dataset using feature selection and speaker normalization. We can see that for emotion classification on noisy data, it is more effective to train the system using noisy data than using clean data. Although speaker normalization helps to combat the overall increase in energy for the noisy data, it does not help with features in the frequency domain. When trained with noisy data, the system can, on the other hand, learn the spectral features for noisy speech. Therefore, we can see from Fig. 3.7 that the decision-level correct classification rate does not drop too much for training and testing on noisy data.

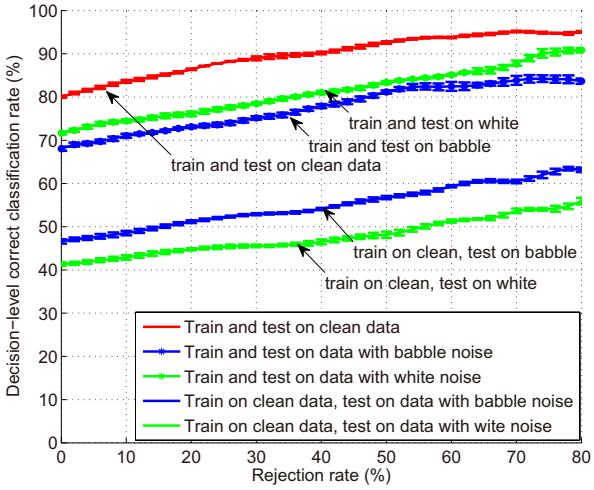


Figure 3.7: Decision-level correct classification rate vs. rejection rate for a general test on clean and noisy LDC data at 5 dB SNR. Speaker normalization, feature selection, and over-sampled training sets are used.

Test on UGA data

We also evaluate our emotion classification system on the UGA dataset [64], in which different emotions are acted by university students. As with the prior tests, cross-validation is performed using the UGA dataset, and the decision-level correct classification rates are shown in Fig. 3.8 for the general test and the gender-dependent tests. Speaker normalization, feature selection, and the over-sampled training sets are used.

Although the UGA dataset contains many more samples than the LDC dataset to train the system, the decision-level correct classification rate for the UGA dataset is decreased from 80% for the LDC dataset to 45% for the UGA dataset for general tests with no data rejected. This drop in performance is mainly due to the fact that the emotion expressed in the UGA data is not very strong and explicit, which makes it hard to effectively train the system. The decision-level correct classification rate increases from 45% to 56% when 50% of the data is

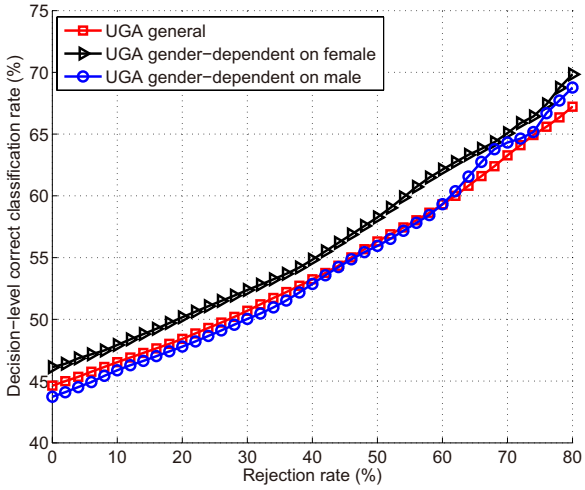


Figure 3.8: Decision-level correct classification rate vs. rejection rate for a general test and gender-dependent tests using the UGA dataset. Speaker normalization and feature selection are used.

rejected as unclassified. We can see that the decision-level correct classification rate is increased by 24.4% of 45% for the UGA dataset when the rejection rate increases from 0 to 50%. Compared with the decision-level correct classification rate for the LDC dataset shown in Fig. 3.5, in which this increase is only 16.2% of 80%, we find that the thresholding fusion mechanism may provide more benefit for more realistic scenarios.

Similar with the gender-dependent results for the LDC dataset, the gender-dependent tests using the UGA dataset provide better results for female speakers than for male speakers. Also, since the UGA dataset contains 133 speakers, speaker normalization becomes important.

Table 3.3: Summary of decision-level correct classification rates (%) for the LDC dataset and the UGA dataset at rejection rates of 0, 50%, and 80%. Speaker normalization and feature selection are used for both the LDC dataset and the UGA dataset. Over-sampled training sets are used for the LDC dataset.

	LDC dataset			UGA dataset		
Rejection rate (%)	0	50	80	0	50	80
General test	80	93	95	45	56	67
Gender-dep. female	87	98	99	46	58	70
Gender-dep. male	76	91	95	44	53	69
Speaker-indep.	47	59	65			
Train white, test white	72	83	91			
Train babble, test babble	68	81	84			
Train clean, test white	41	48	56			
Train clean, test babble	47	57	63			

3.6 Conclusions and Future Work

In this chapter, we present a speech-based emotion classification system based on multi-class SVM and a thresholding fusion mechanism. A full analysis is provided for different test scenarios. A summary of the results are presented in Table 3.3.

Results show that our system outperforms several state-of-the-art methods. Also, the thresholding fusion mechanism is proven to effectively increase the emotion classification accuracy, and the increase is more important for non-professionally acted recordings. In addition, the system performance drops for some more realistic and challenging situations, but the overall results are still acceptable. For emotion classification on noisy data, it is more effective to train the system using noisy data than using clean data.

For future work, we will explore other noise-resilient speech feature extraction methods in addition to F_0 detection using the BaNa algorithm. Also, we are developing an Android implementation to extract the speech features used in this work, which are then sent to an online server for emotion classification. Since

the application only sends the statistics of the speech features to the server for processing instead of the entire speech utterance, the privacy of the user is better preserved and the bandwidth for transmission is reduced. A crowdsourcing test is also on-going using Amazon Mechanical Turk. In this test, users classify the speech utterances from the LDC and UGA datasets, so that we can compare the emotion classification results from the human coders with the results obtained from our system.

4 Context-rich Emotion Classification on Smartphones

4.1 Introduction

Nowadays, smart phones are equipped with a number of sensors: microphone, camera, accelerometer, gyrometer, GPS, etc. People's lives have been greatly eased and entertained with these devices. With biometric sensors embedded, some smart phones can also help monitor a user's physical condition. However, none of these sensors provides information about the user's emotional state. Though emotion classification has been explored extensively in non-mobile contexts, two reasons drive us to implement the emotion classification function on the phone. First, by enabling emotion classification on the phone, people's emotions can be monitored anytime, anywhere. Since people's emotions may change rapidly with different times, places, and people that they are with, the phone can take advantage of the users' mobility to capture people's emotions in real time. Second, emotion is difficult to detect in the sense that people are reluctant to reveal their real emotion to something or someone unfamiliar. Phones, instead, are an everyday companion to the user, and thus have the advantage of capturing the user's real emotion. Note that no facial expressions or body gestures are used in this work for enhancing the emotion classification accuracy.

Therefore, we are motivated to introduce a novel application, which creates an emotion sensor that we call 'listen-n-feel,' deployed on the Windows Phone 7 platform. The application specifies the user's emotion, based on the way that the user speaks to the phone. The speech signal's energy, fundamental frequency (F_0), and other features are analyzed by using speech signal processing methods. By definition, emotion is the complex psychophysiological experience of an individual's state of mind as interacting with biochemical and environmental influences. The proposed 'listen-n-feel' app outputs whether the user is happy or sad.

4.2 Related Work

Speech-based emotion detection technologies have been studied over the years, for different languages. For example, [133] predicts emotion from Chinese speech samples, using F_0, speaking rate, and other speech features, and uses support vector machine for emotion prediction. Recently, a number of emotion detection applications have been developed on mobile platforms. A phone application called 'EmotionSense' [5] was implemented on the Symbian S60 platform. Though the application also detects emotion from speech and no speech content is analyzed, as our scheme does, no speech signal features are explicitly extracted and used as metrics. Instead, a Gaussian mixture model is trained using the entire speech data, thus their approach needs relatively long computing time.

There are also several emotion detection apps on the iPhone and Android online marketplace. However, algorithms employed by these online apps are unknown. The BlackBerry Empathy concept phone detects emotion based on biometric data captured from a special ring, instead of the user's speech. The Siemens CX70 EMOTY phone uses human gestures to do the emotion detection. A 'Mood Meter', developed by Massachusetts Institute of Technology, captures the users' facial expressions to analyze their emotions [134]. Kinect, a motion sensing input

device from Microsoft offers a software development kit with the Windows speech recognition API. However, this only accepts simple verbal commands from the user. No emotion detection function is supported.

4.3 Proposed Application Scenarios

There are a variety of scenarios on which the proposed emotion sensor can make an impact. People participating in Citizen Science projects [135] help scientists collect data in their everyday lives. Sociologists can use people's emotion data to evaluate the city's happiness index, or study the factors that may influence people's emotion.

In the healthcare field, an emotion classification application can be used by doctors (with the patient's permission) to monitor patients with psychosocial concerns, such as depression or anxiety. Healthcare providers can receive real-time assessments, without relying on self-reporting of past events from the patient. An emotion sensor can also be installed on cars, to provide warning to other cars nearby if it detects the driver is angry or aggressive [32]. Note that certain privacy issues should be considered when deploying these sorts of applications.

For game enthusiasts, their emotion revealed from their voice can be reflected on the characters they play in mobile role-playing games. Thus, users may be provided with a more enjoyable user-computer interaction experience.

In social networks, a user can *like* a photo on Facebook by saying 'You look fabulous in this dress!' Though people's comments are typically considered opinionated or subjective instead of expressing a personal state, emotion classification is akin to detecting personal states.

Smart phones may be customized according to users' emotions, such as automatically changing theme settings, or choosing songs based on the user's emotion, making the phone a more personalized and enjoyable gadget.

4.4 System Architecture for 'Listen-n-feel'

4.4.1 Design

The system architecture for the mobile emotion sensor 'listen-n-feel' is shown in Fig. 4.1. The phone captures a user's voice using a microphone, and sends the recorded test sample to a server using Windows Communication Foundation, an API in the .NET Framework. Digital speech signal processing methods are used to extract speech features from the test sample in real time (on line), and extract speech features from a training database off line. We use logistic regression, a linear machine learning method, to train the system off line, as well as to predict the emotion on line, using weights generated during the off-line training process. Finally, the predicted emotion is output back to the phone, as the example shows in Fig. 4.1 'You are in a good mood!' Note that our method only uses speech features for emotion prediction, such as F_0 and energy. In other words, we only predict emotion by investigating how people speak, instead of what people say. Thus, no speech content is used for determining the emotion. In this way, our method can better preserve the phone user's privacy.

4.4.2 Prosody database

The prosody database we use [1] contains speech samples performed by actors and actresses speaking neutral-meaning number and date utterances (e.g., 2001, December 8) with different emotions. There are 14 emotion categories in total in the database, including boredom, cold anger, contempt, despair, disgust, elation, happy, hot anger, interest, panic, pride, sadness and shame. In our work, speech utterances with happy, interest, elation and pride emotions in the database are used as happy emotion training samples, and speech utterances with sad, bored, shame and despair emotions are used as sad emotion training samples. The reason

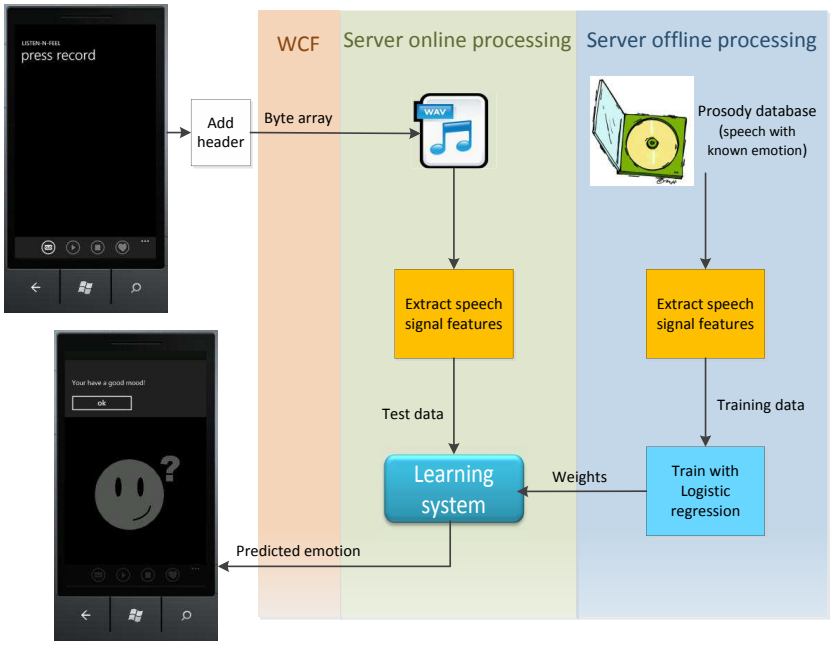


Figure 4.1: Illustration of the emotion sensor system architecture.

why we only focus on a broad happy emotion and a broad sad emotion classification is that these two broad emotion categories are easy to differentiate, since the happy emotion is active and positive, while sad is passive and negative, as defined in [97]. Thus these two broad emotion categories are suitable for preliminary test of the proposed app. Also, we have more training samples for happy and sad by considering samples of four emotions for happy, and samples of four emotions for sad. Thus, the machine learning model will be better trained, and will provide a higher prediction accuracy. We envision that in future versions of the app, richer emotion categories will be supported, as the precision of the proposed app become higher. Thus, emotions like angry will be included, and we can differentiate interest and elation, instead of considering both of them as a broad happy emotion.

4.4.3 Speech features

In the database, every speech utterance is about one to two seconds long. We first do noise reduction using a 50 to 4000 Hz bandpass filter to suppress the noise outside this frequency band. Then we separate every utterance into short segments of 40 ms. Since human speech consists of audible and inaudible segments, we only analyze speech features for audible segments, and ignore the inaudible ones. Therefore, computation time and power are saved by skipping the inaudible segments. The frame energy is used to differentiate audible and inaudible segments.

Digital signal processing methods are used on each audible segment to extract speech signal features. We choose 12 features that are frequently used in emotion recognition [136], and are explained below.

- Fundamental frequency (F_0): F_0 is defined as the relative highness or lowness of a tone as perceived by the ear. It depends on the number of vibrations per second produced by the vocal cords. We use autocorrelation in the time domain to extract F_0 values.

- F_0 difference: the difference of F_0 values between two neighboring segments. This feature depicts the changing of F_0 values. More fluctuations of F_0 values may indicate active emotions, such as happy and angry.

- Energy: the loudness of speech. We calculate the energy for each segment by taking the summation of all the squared values of the samples' amplitudes.

- Energy difference: the difference of energy values between two neighboring segments. Similar to F_0 difference, more variations may indicate active emotions.

- Frequency and bandwidth for the first four formants (8 features): formants are determined by the shape of the vocal tract, and are influenced by differ-

ent emotions. For example, formant bandwidth during slackened articulated speech is gradual, but during improved articulated speech is narrow. We use linear predictive coding to calculate frequency and bandwidth for the first four formants [136].

With the knowledge of 12 feature values of each segment, we can calculate 6 statistic values of each feature, thus $12 \times 6 = 72$ metrics in total are derived for each speech utterance file. The 6 statistic values we consider are: mean, max, min, range, standard deviation, and median.

4.4.4 Emotion prediction

As shown in Fig. 4.1, we use logistic regression, a linear machine learning method, to train the system as well as to predict emotions. This method predicts the probability of occurrence of an event by fitting data to a logistic curve. In this linear model, the 72 metrics are used as predictor variables, and the weights for each metric are used as regression coefficients. By training the system with the prosody database, the weights for each predictor variable are determined. Thus, given the 72 metrics of the test speech sample and the weights from the trained logistic regression model, we can predict the emotion for the test speech sample. As shown in Fig. 4.1, the server will output the predicted emotion back to the phone at the end.

4.4.5 System implementation

The implementation of the entire system consists of several components. The extraction of 12 speech signal features is done using MATLAB. The logistic regression algorithm and the Windows Phone 7 app user interface are implemented in C#. The 'listen-n-feel' service is hosted on the cloud using Internet Information

Table 4.1: Confusion matrix for broad happy and sad emotions using cross-validation.

emotion	**happy**	**sad**
happy	283	129
sad	120	315

Services. Finally, the phone-server communication uses Windows Communication Foundation. Some other issues should also be addressed to guarantee a smooth and reliable server-phone communication, such as firewall configuration and access control settings. Finally, the 'listen-n-feel' app is physically deployed on a Samsung Focus Windows phone, and works with 3G or WiFi connections to a server.

4.5 Evaluations

To evaluate the performance of the 'listen-n-feel' emotion sensor, we conduct cross-validation tests on one standard prosody database [1]. First, we use cross-validation to evaluate the prediction accuracy of the emotion sensor. Multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds. There are 847 speech utterances from the database, with 403 happy utterances and 444 sad utterances. Each speech utterance is around 1-3 seconds long. Table 4.1 shows the confusion matrix, with an emotion classification accuracy of (283+315)/(283+315+120+129) = 70.6%.

Table 4.2 lists the top five largest absolute values of weights among all 72 metrics, from which we can see that the energy and F_0 values have the highest weights during the decision making process, and thus strongly influence the prediction outcome. The sign of the weight denotes whether the corresponding metric

Table 4.2: Weight values for speech signal feature metrics.

top order	**metric**	**weight**
1	mean F_0 difference	0.699
2	mean energy difference	-0.672
3	min energy	-0.273
4	standard deviation energy difference	0.245
5	range energy	0.159

makes a positive or negative influence on the prediction outcome.

4.6 Conclusions

Since the proposed emotion sensor only detects emotion based on speech prosody features, and no speech content is analyzed, people can fake their speech when we ask them to say something to the phone. For example, one user said 'I have some good news for you' in a very flat tone with low energy. It turned out that the phone predicted the emotion of the user to be sad, instead of happy. Therefore, we will sample snippets of speech from users' regular phone calls, to catch the most reliable samples for emotion prediction.

Prosody features differ from person to person, which affects the prediction accuracy when comparing the user's voice with a standard prosody database. Therefore, we believe that in the future, we can further improve the accuracy by first training the emotion sensor with the users voice. This will allow the phone to be adapted to the user's F_0 and energy range, and the normal level, for instance. Another issue for real deployment of this mobile emotion sensor is the emotion prediction latency, which highly depends on the length of the utterance and on the WiFi or 3G transmission conditions. Longer utterances require longer time

for both signal computation and audio file transmission.

In our work, we do all the computation of emotion detection on the server, but it is also possible to distribute the computation burden to both the server and the phone. We qualitatively compare the cost of computation speed, power consumption, and data transmission speed between these two approaches.

If we do all the computation on the server, computation-intensive tasks like emotion detection can benefit from fast computation by the server and lower power consumption on the phone, thus phone battery is saved. However, we need to transmit the entire recorded wave file from the phone to the server for processing, thus data transmission takes longer time.

For the other approach, we can do the signal processing part on the phone, i.e., extracting speech signal features, and only transmit statistic values of speech features to the server. This approach can achieve a shorter data transmission time, but suffers from slow computation as well as high energy consumption on the phone.

Part II

Energy Efficient Wireless Sensor Networks

5 Cross-layer Energy Optimization Under Image Quality Constraints for Wireless Image Transmissions

5.1 Introduction

Wireless image transmission is important for a variety of applications, from security and surveillance to in-home monitoring. However, wireless imagers are battery-operated, and hence the energy-efficient design of these systems is critical to their operation. Most existing studies on energy optimization of wireless communication consider error-free packet transmission, where the entire packet has to be retransmitted if there is even a single bit error. However, for image transmission applications, there is often a certain tolerance to errors in the received data, as errors in the decoded data become distortion in the image content. In this chapter, we derive a cross-layer model of the energy consumption of wireless imagers. Using this model, we can jointly optimize the transmit power at the physical layer, and packet retransmissions and the packet length at the MAC layer, while meeting an image quality constraint issued by the application layer.

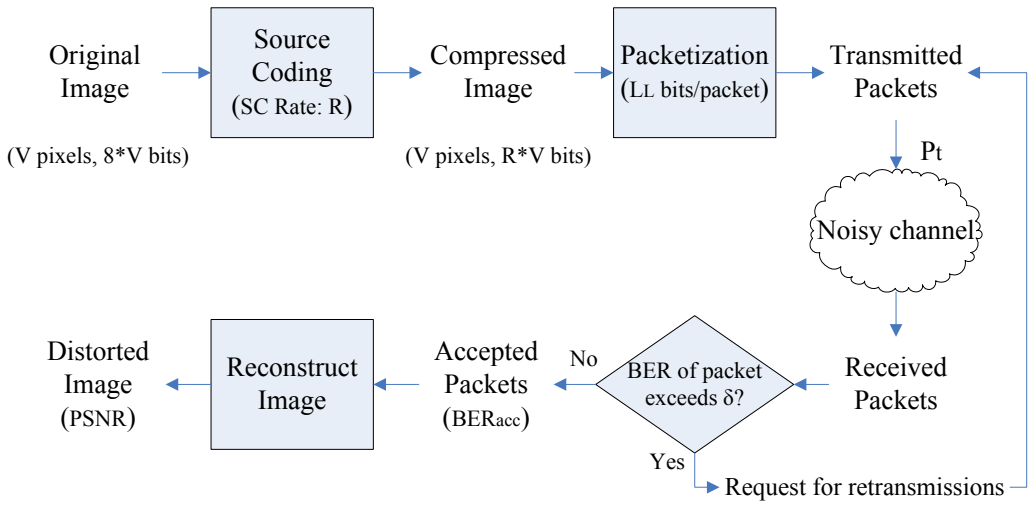


Figure 5.1: The image compression, transmission and reception processes.

Consider the following image transmission procedure: after source coding an image, the resulting information bits are packetized based on a *packet length*, and then sent to the receiver with a specific *transmit power*. The receiver will determine the error rate in each received packet. We define an internal system parameter δ to represent *the packet-level error-tolerance rate*. If the error rate of a packet exceeds δ, a retransmission of the packet is requested. The image constructed at the receiver side using the accepted packets is required to have a higher image quality than the image quality constraint issued by the application layer. Fig. 5.1 shows the image transmission process.

Considering this procedure, we propose a system model that determines the energy per image as a function of the distance to the receiver, the packet length, and the transmit power. Additionally, using the model in [137] we can relate the packet-level error-tolerance rate to the peak signal-to-noise ratio (PSNR) of the reconstructed image. Note that for a coded image like JPEG that has dependencies, an error in the coded data may cause different quality degradation if it occurs at different locations in the data. Thus, we only consider the average PSNR for

all received images, as done in [137], taking into consideration different quality degradations. Given the energy model and the image quality degradation model, we can determine the optimal packet length and transmit power that minimize the energy per image given an application-level PSNR constraint on the reconstructed image.

5.2 Related Work

Although there are several studies to minimize the energy consumption with transmit power control [138], or packet length optimization [139], or based on a distortion constraint [140, 141], our work is the first one to consider all of these design criteria to derive a cross-layer joint optimization, and an internal packet-level error-tolerance parameter is defined to relate the image quality constraint to packet-level optimizations.

In [142], the authors consider the joint optimal design of the physical, MAC, and routing layers to maximize the lifetime of energy-constrained wireless sensor networks. However, their work does not consider the impact of the application layer on energy saving. Our work takes advantage of the user-defined image quality constraint, making the energy minimization problem more flexible while satisfying the application level requirements.

The joint energy optimization problem in terms of adjusting both packet length and target bit error rate (BER) has been studied in [8], where the authors assume that the data must be received error-free. However, this restriction is not required for multimedia applications, where a user's requirements may vary. Thus, it is imperative to also consider the application level quality requirement and relax the error-free transmission constraint in the optimization.

In [140], image quality is evaluated in terms of distortion observed in the application layer. The authors build a comprehensive power-rate-distortion optimiza-

tion framework for image sensors. However, in that framework, retransmissions are not considered. If an intra or inter macroblock is lost, the previous macroblock is repeated as the current one. Also, users do not have control over the received image quality. In our work, on the other hand, retransmissions are considered, and the users can define their own received image quality requirement.

Joint optimization of the application layer, data link layer, and physical layer is studied in [143] using an application-oriented objective function in order to maximize user satisfaction. However, our goal is to reduce the total energy consumption instead of maximizing the user satisfaction.

5.3 Cross-layer Energy Consumption Modeling and Energy Minimization

In this section, we present the cross-layer energy minimization problem for the communication of images over wireless links. There are three controllable parameters we consider: the payload length L_L, the transmit power P_t, and the packet-level error-tolerance rate δ. As we will show in this section, the energy consumed to transmit an image depends on these parameters and the transmission distance d. Our goal is to minimize the energy consumption per image by finding the optimal values for these three controllable parameters to meet a given PSNR threshold, $PSNR_g$, for a given transmission distance, d_g. Also, in practice, the payload length is limited by different factors, such as application level data rate, latency constraints, and the allowable packet length limits introduced by the MAC layer. Accordingly, we define the maximum value of a payload length to be

L_{max}. The optimization problem can then be expressed as

$$\begin{aligned} \min_{\{P_t, L_L, \delta\}} \quad & \overline{E}_{image}(P_t, L_L, \delta, d), \\ s.t. \quad & PSNR(P_t, L_L, \delta, d) \geq PSNR_g, \\ & d = d_g, \\ & 0 < L_L \leq L_{\max}, \\ & P_t > 0, \\ & \delta \geq 0, \end{aligned} \tag{5.1}$$

where $\overline{E}_{image}$ is the average total energy for successfully transmitting and receiving one image, and $PSNR_g$ is the given PSNR threshold of the system, which denotes the requested received image quality. In the following sections, we derive the mathematical formula for $\overline{E}_{image}$, and we present the relationship between image quality and the communication distortion.

5.3.1 Packet structure and energy consumption model

The packet structure consists of payload, upper layer header, PHY/MAC-header, and preamble, the size of which are L_L, L_{UH}, L_H and L_P bits, respectively. In this chapter, we assume that the entire packet is modulated using uncoded Binary Phase Shift Keying (BPSK) and that the transmit power is constant during the entire packet. Also, the upper layer header, PHY/MAC-header and preamble are assumed to be protocol-dependent fixed length components. Hence, we only need to optimize the length of the payload.

We look at minimizing the total energy consumption required per one image. Let V denote the image size in pixels, L_L the payload length, and R the source coding rate in bits per pixel (bpp). The number of packets forming the image is, then, $\frac{V \cdot R}{L_L}$. If this value is not an integer, the remaining data is packetized into a packet of payload size less than L_L.

If we let $\overline{E}_{pkt}$ denote the average energy consumption for successful transmission of a packet, the total energy for successfully transmitting and receiving one image can be expressed as $\overline{E}_{image} = \frac{V \cdot R}{L_L} \cdot \overline{E}_{pkt}$. There are four energy components of $\overline{E}_{pkt}$: transmission energy E_{tx}, transmitter circuit energy E_{ct}, receiver circuit energy E_{cr}, and source coding energy E_{sc}. Let P_{ct} and P_{cr} represent the power for the transmitter circuits and receiver circuits, respectively, and $T_{on}(L_L)$ denote the time the nodes must remain in the *on* state for one successful or unsuccessful transmission of a packet. Then, the three communication energy components are formulated as

$$E_{tx} = P_t \cdot T_{on}; \;\; E_{ct} = P_{ct} \cdot T_{on}; \;\; E_{cr} = P_{cr} \cdot T_{on}, \tag{5.2}$$

where P_t is the transmit power. Note that all three energy components are functions of T_{on}, and hence, of L_L. We assume the source coding energy, E_{sc}, is constant and cannot be optimized, for the sake of simplicity.

We consider a point-to-point transmission over an additive white gaussian noise (AWGN) channel with no fading or interference from other transmissions. For AWGN channels, the bit error rate P_b can be approximated as a function of transmit power P_t and distance d:

$$P_b(P_t, d) = \frac{1}{2} e^{-\frac{P_t}{2BN_0} \cdot \left[\frac{\sqrt{G_l}\lambda}{4\pi d}\right]^{3.5}}, \tag{5.3}$$

where we assume a path loss exponent of 3.5 and B is the signal bandwidth, N_0 is the noise power spectral density, G_l is the product of the transmit and receive antenna gains, and λ is the wavelength.

The *on* time of nodes is calculated as $T_{on} = T_L + T_{UH} + T_H + T_P = \frac{1}{R_b} \cdot (L_L + L_{UH} + L_H + L_P)$, in which T_L, T_{UH}, T_H and T_P refer to the transmission time for the payload, upper layer header, PHY/MAC layer header and preamble, respectively. R_b is the bit rate and equals to the signal bandwidth B for uncoded BPSK modulation. Therefore, T_{on} is a function of our optimization parameter L_L.

In the case that a packet needs to be retransmitted, the energy consumption for data transmission and transceiver circuits power should be multiplied by n, i.e., the total number of transmissions (including the first transmission). However, we do not need to multiply E_{sc} by n, since we do source coding to the original image once before transmission. Therefore, if the number of transmissions for a single packet is n, the total energy is

$$E_{pkt}(n) = n \cdot T_{on}(L_L) \cdot [P_t + P_{ct} + P_{cr}] + E_{sc}. \tag{5.4}$$

For the *error-tolerance control*, we define a packet-level error-tolerance rate variable, denoted as δ. This internal system parameter is adjusted based on the *allowed image quality level* assigned by the application layer. Considering the commonly employed selective repeat ARQ in our model, if the bit error rate of a received packet is less than or equal to δ, then the packet is accepted and passed to the upper layers. Otherwise, the received packet will be discarded, and a NACK is sent to the transmitter to request a retransmission of the packet. Mathematically, the maximum number of errors in a single packet that can be tolerated at the receiver is then $\lfloor (L_L + L_{UH})\,\delta \rfloor$, where $\lfloor \bullet \rfloor$ is the floor function. All packets with less than or equal to $\lfloor (L_L + L_{UH})\,\delta \rfloor$ errors are accepted, thus the probability of accepting a received packet is the summation of all the possibilities of having less than or equal to $\lfloor (L_L + L_{UH})\,\delta \rfloor$ errors, which can be formulated as:

$$Pr_{acc} = \sum_{i=0}^{\lfloor (L_L+L_{UH})\delta \rfloor} C^i_{L_L+L_{UH}} (1-P_b)^{L_L+L_{UH}-i} P_b^i, \tag{5.5}$$

where i is the number of errors in the packet.

As stated above, n denotes the total number of transmissions for the successful reception of a packet. The expected value of n, N, is then

$$N = E[n] = \sum_{n=1}^{\infty} n \cdot Pr_{acc}(1-Pr_{acc})^{n-1} = \frac{1}{Pr_{acc}}. \tag{5.6}$$

Therefore, by taking the expectation of E_{pkt} given in (5.4), the average total energy consumption is

$$\overline{E}_{image} = \frac{V \cdot R}{L_L} \cdot \{N \cdot T_{on} \cdot [P_t + P_{ct} + P_{cr}] + E_{sc}\}. \tag{5.7}$$

Plugging (5.3), (5.5) and (5.6) into (5.7), we obtain the total energy to transmit an image as a function of transmit power, P_t, packet payload length, L_L, transmission distance, d, and the packet-level error-tolerance rate, δ.

5.3.2 Mathematical relationship between error-tolerance rate and received image quality

In the previous section, we modeled the average total energy per image $\overline{E}_{image}$ with a tolerable error rate δ. In this section, we further study the impact of δ, and the communication distortion on the received image quality.

While δ is the error-tolerance threshold per packet, it is not a performance metric and does not evaluate the overall quality of the received data directly. On the other hand, the average bit error rate of all *accepted* packets, BER_{acc}, does. Given an overall accepted packet BER, BER_{acc}, we need to consider how the randomly distributed channel bit errors in the received image influence the overall image quality. In this chapter, we investigate JPEG compressed images as the application source, for which a joint source-channel distortion model is presented in [137]. The authors derive the overall image distortion caused by quantization during source coding and by channel bit errors, assuming 8-bit unsigned representation for unquantized pixel values. Also, data is in the form of a stream in [137], and not packetized. Thus, P_b is directly used to calculate the channel errors. However, in our study, we use retransmissions to guarantee the received packets' quality in terms of bit error rate in the packet. Thus, at the receiver end, we use BER_{acc} instead of P_b to calculate the impact of random bit errors that are caused

by the channel on the received image quality, which is different from [137]. We fix the source coding rate R to 1.25 bpp and only focus on the impact of channel distortion on the image quality.

To use the PSNR image quality model presented in [137], we need to derive the average bit error rate of all *accepted* packets BER_{acc}. Let the variable i denote the number of errors in a received packet. The average BER among all accepted packets is then

$$\begin{aligned} & BER_{acc}\left(P_t, L_L, \delta, d\right) \\ & = \frac{\sum_{i=0}^{\lfloor (L_L+L_{UH})\delta \rfloor} i \cdot C^i_{L_L+L_{UH}} \left(1-P_b\right)^{L_L+L_{UH}-i} P_b^i}{\left(L_L+L_{UH}\right) \cdot Pr_{acc}}, \end{aligned} \tag{5.8}$$

where the numerator is the average number of errors for accepted packets, which is normalized by the packet acceptance probability Pr_{acc} in the denominator.

By plugging (5.7) into (5.1), the energy minimization problem formulation is derived, where PSNR is a function of BER_{acc} (defined in (5.8)) as presented in [137].

5.4 Energy Optimization Results

In this section, we obtain the optimal P_t, L_L and δ tuple, i.e., (P_t^*, L_L^*, δ^*) for the energy optimization problem in (5.1). The importance of such an optimization is presented by comparing the default behavior and the optimized behavior of a ZigBee commercial mote and a WiFi commercial mote. The investigated modules are Crossbow's MICAz mote [144] and Microchip's ZG2100M/ZG2101M transceiver module [145], respectively. Since optimization parameter L_L is an integer, the resulting mixed integer programming problem is not a convex problem. Thus, standard methods like the Karush-Kuhn-Tucker approach or Lagrange multipli-

ers cannot be used to obtain the optimal solution. Instead, we find the optimal solutions by using numerical optimizations in MATLAB.

5.4.1 Numerical evaluation setup

To evaluate the effect of the optimization parameters on the performance of a real life communication system, we first define the communication parameters to be the same values as those used in the MICAz motes [144], but keep the transmit power P_t and the payload length L_L as variables. The carrier frequency is $f_c = 2.4$ GHz, the transmit data rate is $R_b = 250$ kbps, the upper layer overhead length for ZigBee is $L_{UH} = 160$ bits, the PHY/MAC layer header length is $L_H = 32$ bits, the preamble length is $L_p = 32$ bits, and the maximum payload length is set to $L_{max} = 10^5$ bits, which is 12.5 KB. The transmit circuit power is set to $P_{ct} = 11mA \times 3V = 33\,mW$, which is the lowest power level used for data transmissions according to the MICAz specifications. Similarly, the receiver circuit power is set to $P_{cr} = 19.7mA \times 3V = 59.1\,mW$. Since the energy consumption for source coding E_{sc} is assumed to be a small constant compared to the communication energy, and it does not affect the results for our optimization problem, we simply neglect it. We assume that the transmitted image size is $V = 512 \times 512$ pixels, and fix the source coding rate to $R = 1.25$ bpp.

Based on the distortion model given in [137], $PSNR$ is determined by BER_{acc} and R. Since we fix the value of R to 1.25 bpp, $PSNR$ is only determined by BER_{acc}. Therefore, we can change the PSNR inequality constraint to a BER_{acc} inequality constraint as $BER_{acc} \leq BER_{acc}^{g}$, where BER_{acc}^{g} denotes the BER_{acc} value for $PSNR_g$ at $R = 1.25$ bpp found from [137].

The numerical minimization of the energy per image is done for a given image quality and transmission distance, and on a given range of the payload length L_L, the transmit power P_t and the error-tolerance rate δ. The investigated error-

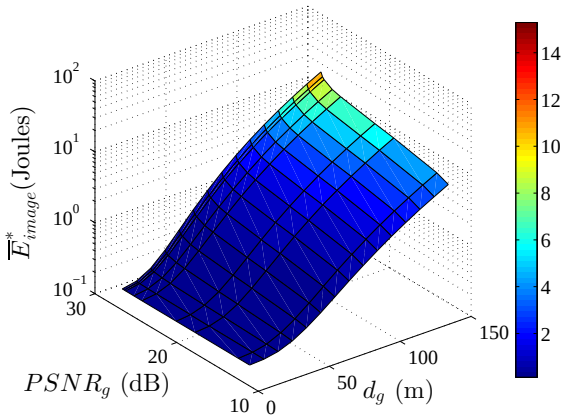


Figure 5.2: Minimum energy per image for varying d and PSNR threshold values.

tolerance rate interval is set to be from 10^{-7} to 10^{-1}. The search interval for an optimal payload length L_L is from 10 to 10^5 bits since $L_{max} = 10^5$ bits. Because the transmit power P_t is a function of P_b and d (see (5.3)), instead of running the brute-force search on a range of P_t values, we run on P_b values and then calculate the optimal P_t from the optimal P_b. The searching range for P_b is 10^{-7} to 10^{-1}. We investigate the effects of different transmission distance values by varying d from 10 m to 150 m. The PSNR value ranges from 12.8 dB to 28.4 dB, according to the available PSNR value range given in [137].

5.4.2 Numerical solution for the energy optimization problem

In this subsection, we numerically find the optimal (P_t, L_L, δ) tuple, i.e., (P_t^*, L_L^*, δ^*), that yields the minimum energy per image, $\overline{E}^*_{image}$. We present the minimum energy values found for different PSNR thresholds and transmitter-receiver distances, d, along with the corresponding optimal system parameter values yielding the minimum energy.

The minimum energy per image $\overline{E}^*_{image}$ values achieved for different PSNR thresholds and transmitter-receiver distances, d, are shown in Fig. 5.2. As revealed

in the figure, the minimum energy per image increases as the transmitter-receiver distance increases, which is expected since higher transmit power is needed to compensate for the larger path loss. Additionally, we can observe that the minimum energy per image also increases as $PSNR_g$ increases, i.e., if the user requires a better image quality, more energy is dissipated. We find that the optimal transmit power has a similar trend (not shown).

The optimal packet length, L_L^*, is found to always be the maximum possible value within the allowable range, i.e., L_{max}. The reason is that since the error-tolerance rate δ defines the ratio of the number of erroneous bits to L_L, as L_L increases, the number of allowable erroneous bits also increases. Thus, the effect of the packet overhead decreases. As a result, the higher the L_L value, the lower the energy per image. We also find that the optimal probability of error, P_b^*, decreases as $PSNR_g$ increases, since a better received image quality requires a lower bit error rate over the channel.

The simulation results show that the number of transmissions is very close to one for all optimal cases, which means that we just need to transmit the image once with appropriately configured parameters, and the received image quality will, on average, meet the user-defined PSNR requirement. The reason is found to be that retransmissions are a very energy-inefficient way to enhance the received signal quality. Simulations show that the P_b^*/δ^* ratio is always smaller than 1, which means that the error-tolerance rate at the receiver end is larger than the average BER over a noisy channel. Therefore, we have no retransmissions on average.

With the numerical solutions derived for the energy optimization problem, we can set P_t, L_L and δ values according to the user-defined PSNR requirement. However, due to the fact that we do not have retransmissions for optimal cases, we accept every received packet. Hence, there is no need to use sophisticated error detection codes to compute the number of errors in each received packet to

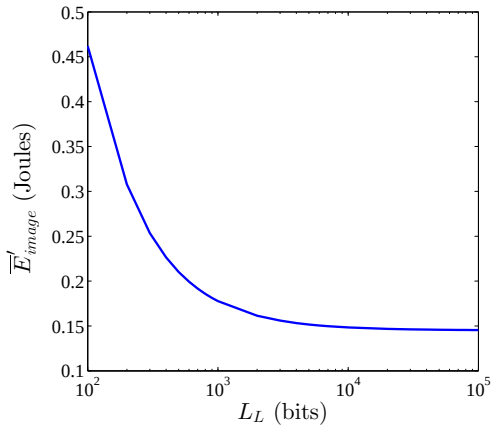
decide whether it needs to be retransmitted. Yet, δ is an important intermediate variable of the optimization process, and used to determine the values for optimal P_t and optimal L_L.

5.4.3 Sensitivity analysis of the optimization parameters

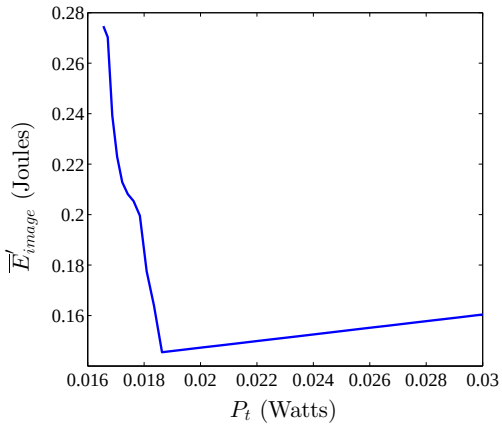
In this subsection, we investigate how sensitive the *mean energy per image* is to the P_t, L_L and δ values, and the importance of choosing the optimal P_t, L_L and δ values to save energy. For all evaluations in this subsection, the PSNR threshold is set to 19.5 dB and $d = 30$ m. For each of the three analyzed parameters, we evaluate the energy performance by fixing one of the parameters while optimizing the other two parameters. We denote the resulting energy per image as $\overline{E}'_{image}$, since it does not represent the overall minimum energy value, which is derived by optimizing all three parameters. For example, to show the sensitivity of $\overline{E}_{image}$ to L_L, we plot $\overline{E}'_{image}$ at and around L_L^*, as shown in Fig. 5.3a. As expected, $\overline{E}'_{image}$ has its minimum value at $L_{max} = 10^5$ bits. $\overline{E}'_{image}$ decreases monotonically as L_L increases, since the longer the packet, the smaller the overhead portion is, and thus the system is more energy-efficient.

Fig. 5.3b shows that $\overline{E}^*_{image}$ lies at $P_t^* = 18.6$ mW. When P_t deviates to a larger value from P_t^*, $\overline{E}'_{image}$ increases, but very slowly. Since the PSNR threshold can be met by transmitting once using the optimal power, using a higher power is unnecessary. When P_t deviates to a lower value from P_t^*, $\overline{E}'_{image}$ increases dramatically, mainly due to the rapidly increasing number of retransmissions.

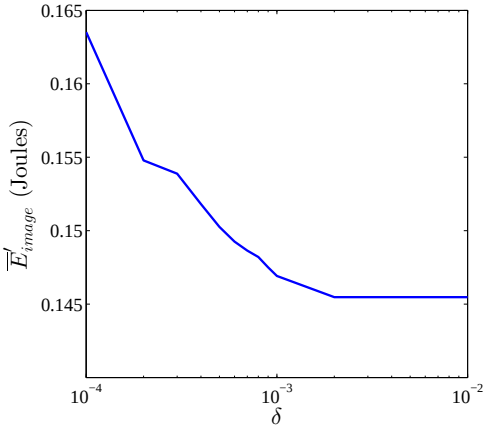
As depicted in Fig. 5.3c, $\overline{E}'_{image}$ decreases as δ increases and remains the same when δ is greater than 0.002. This is because when the error-tolerance threshold is as loose as 0.002, almost every received packet is accepted. Thus, we have $P_b = BER^g_{acc}$. When the error-tolerance threshold becomes looser, since P_b cannot be any larger due to the PSNR constraint, a larger δ has no effect on the received



(a)



(b)



(c)

Figure 5.3: The sensitivity of $\overline{E}_{image}$ to a) L_L, b) P_t, and c) δ.

packet quality. Hence, with constant P_b and L_L values and $N \cong 1$, $\overline{E}'_{image}$ is also constant when δ is greater than 0.002. Thus, all δ values equal to or greater than 0.002 are optimal values. Recall that we define δ^* as the smallest value that gives the minimum energy.

5.4.4 Performance gain over error-free transmissions

In our approach, the received image quality is determined by both source distortion and channel distortion. Thus, it is necessary to compare the energy that our approach consumes with that consumed by error-free transmissions proposed in [8], under the same received image quality constraint. Due to different study criteria or problem settings, we cannot compare our work with other works mentioned in the Related Work Section. In [8], the image quality is only effected by source distortion. They minimize energy by optimizing L_L and P_b. For our approach, one PSNR value is mapped to several (P_b, R) tuples, as stated in [137]. Thus, we minimize energy by optimizing L_L, δ, and the (P_b, R) tuple. Fig. 5.4 compares the minimized energy per image under different image quality constraints. Our approach saves around 10-20% of the energy for middle to large distances. Thus, to achieve the same image quality, distributing distortion in both the source coding and transmission processes achieves lower energy than only allowing distortion in the source coding process.

5.4.5 Performance gain over ZigBee and WiFi systems with default parameters

The potential gains of using the proposed cross-layer joint optimization is quantified by comparing its results to the default MICAz mote parameter settings, provided in [144]. The reference case defines a fixed transmit power of $P_t^{\dagger} = 19.2mW$, and a fixed packet length of 128 bytes. For the reference case, with a fixed L_L and

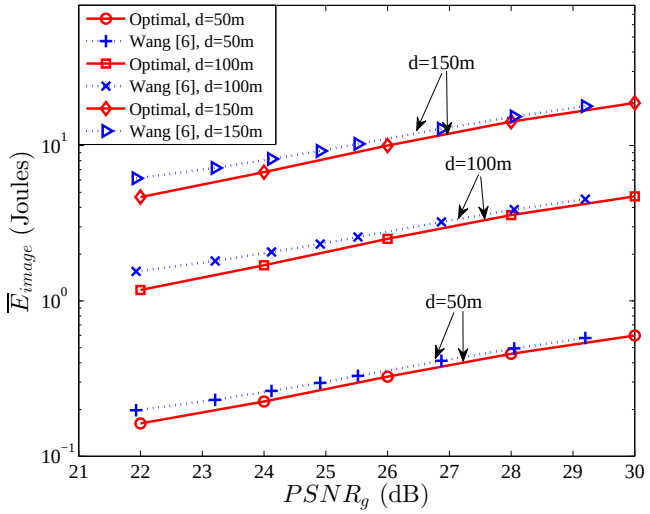


Figure 5.4: Energy performance comparison with [8].

P_t, δ is the only parameter that can be adapted to meet the PSNR constraints. We compare the energy consumption between the optimal case and the reference case for a sample PSNR requirement of 28.4 dB. As seen in Fig. 5.5, at short transmission distances, energy consumption for the optimal case increases slightly as d increases, due to the slight relative increase in transmit power compared with circuit power. For the reference case, energy consumption remains at a constant higher level, since with a fixed transmit power and a fixed packet length, energy per packet is also fixed. Therefore $\overline{E}_{image}$ is solely determined by the expected number of retransmissions. For small d, the reference P_t value is large enough, resulting in no retransmissions and thus a constant $\overline{E}_{image}$. At short distances, our optimized approach saves about 35% of the energy compared to the MICAz motes. At large transmission distances, $\overline{E}_{image}$ for the optimal case is still relatively small, while $\overline{E}_{image}$ for the reference case increases sharply. This is because at large distances, P_b easily becomes larger than the error-tolerance threshold for a fixed P_t. Thus, an enormous number of retransmissions are needed.

Compared with ZigBee motes, WiFi modules normally use higher data rate, larger transmit power, and longer packets. We set all parameter values to be the

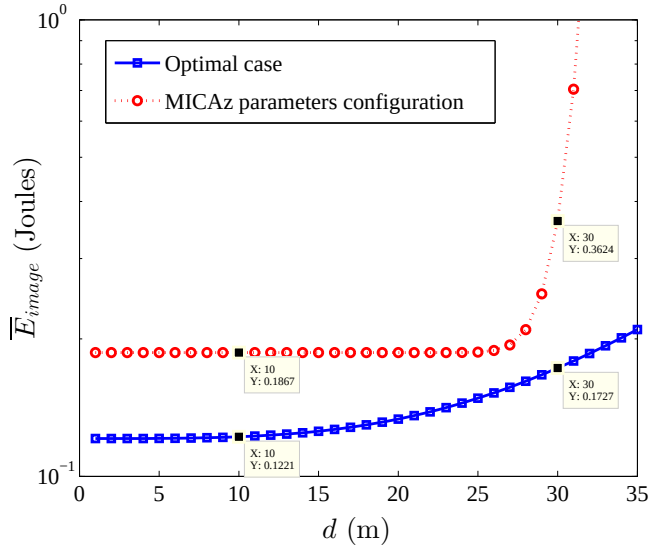


Figure 5.5: Energy consumption comparison with MICAz ZigBee motes at different transmission distances, for PSNR = 28.4 dB.

same as the Microchip motes [145] when conducting the performance comparison. The transmit data rate is $R_b = 2$ Mbps. The transmit circuit power is $P_{ct} = 379.5\,mW$, and the receiver circuit power is $P_{cr} = 280.5\,mW$. The Microchip motes use a fixed transmit power of $P_t^{\dagger} = 128.7\,mW$, and a fixed payload length of 2048 bytes, giving $L_L^{\dagger} = 16,384$ bits. Though using a higher transmit power, the WiFi module consumes less energy than the ZigBee mote to transmit the same size image, since the circuits consume less energy by running for a shorter time. Due to space limitations and a similar performance gain behavior, the WiFi performance comparison figure is not shown. At short distances, around 18% of the energy can be saved with our optimization framework. More importantly, a significant amount of energy can be saved for distances larger than 25 m using our optimization approach.

5.5 Conclusions

This chapter investigates cross-layer energy minimization given image quality constraints for wireless image transmissions. We formulate an energy consumption model with tolerable channel bit errors and retransmissions, and then provide the optimal system parameters to minimize the energy consumption while meeting a certain application-level received image quality constraint. Simulation results show that the minimized energy per image increases as transmission distance and image quality requirements increase. To achieve the same image quality, distributing distortion in both the source coding and transmission processes achieves lower energy than only allowing distortion in the source coding process. Finally, our cross-layer optimization approach is shown to significantly reduce the total energy consumption compared with a ZigBee commercial mote, saving around 35% of the energy at short distances, and much more at middle-to-large transmission distances. Also, our approach saves 18% of the energy compared with a WiFi commercial mote for short distances, and significant energy for middle-to-large distances.

For in-home monitoring systems, energy consumption during the image capturing process also accounts for a large portion of the total energy consumption of the system. Therefore, we explore an energy-efficient in-home video monitoring strategy through sensor placement optimization in the next chapter.

6 Motion Sensor and Camera Placement Design for In-home Wireless Video Monitoring Systems

6.1 Introduction

Nowadays, there is an increasing need for *video monitoring systems*, for applications ranging from surveillance and intruder detection to elderly and infant monitoring. For motion detection systems that utilize motion sensors alone, wireless systems are the preferred choice due to their easy deployment, less cluttered view and the flexibility they provide to be mounted at any location. However, for video monitoring systems, due to the energy hungry operation of cameras, it is difficult to achieve the long lifetimes necessary when the video cameras are wireless and battery-operated. In this chapter, we investigate a video monitoring system that consists of a combination of motion sensors, cameras and motes for communication. For this system, we determine the optimum deployment of these components for different optimization objectives, including maximum lifetime of the system and minimum monetary cost of the system.

The camera placement problem is largely related to coverage, as studied in [146–148]. The authors in [146] use both directional and omnidirectional cameras to cover a specific area. Their objective is to find the minimum number of cameras used, but we will show the benefits to prolonging network lifetime when using more cameras. In [147], the authors solve the placement optimization problem for four different objectives, considering coverage and cost. However, the cameras they use have plug-in power, whereas we consider battery-powered cameras, thus energy consumption is addressed in this chapter. Also, our motion detection sensors can be deployed separately from the cameras in the detached scenarios we evaluate, which brings more flexibility to the placement problem. The study in [148] shows the camera placement problem with different floor plans. However, they only target the coverage problem. Some other work focuses on the energy perspective: [149] presents a multi-tier camera sensor network, where detection, recognition and tracking are performed on different types of sensor nodes and cameras. Our approach is based on a single tier network, and lifetime is considered in addition to the total energy. Camera scheduling and energy allocation schemes to maximize network lifetime are proposed in [150]. However, the authors assume that the cameras are placed randomly and cannot be optimally deployed.

6.2 Video Monitoring System Overview

We assume that the target area that is monitored is a rectangle, with length L and width W. For the sake of simplicity, we only consider the placement problem at the edges of a 1.5m-high flat plane, and assume that the cameras look into this plane, as shown in Fig. 6.1. On this plane, we model the motion sensors' and the cameras' coverage as sectors, with angle of field of view (FoV) θ_m and θ_c, and range R_m and R_c, respectively. We consider three placement options, where i) motion sensors are attached to cameras, and ii) motion sensors are placed separately, and

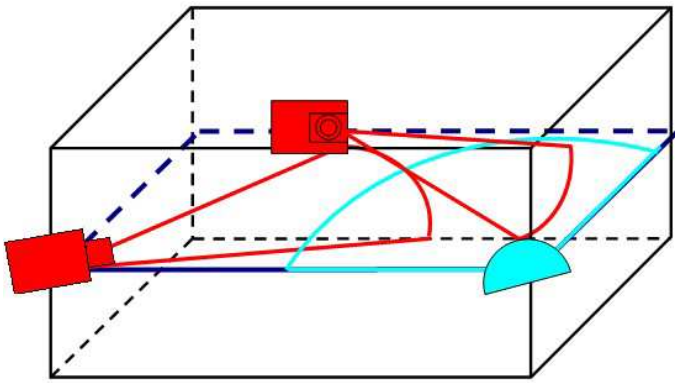


Figure 6.1: Illustration of two cameras (camera-shaped blocks) and one motion sensor (semicircle) placed on a flat plane.

iii) a hybrid approach, where any motion sensor can be attached to a camera or detached from the cameras. If a motion sensor is attached to a camera, it will trigger the camera immediately upon detecting an event; if they are mounted separately, the motion sensor will use a transceiver to transmit a wake-up signal, which is received by the transceiver of the cameras and wakes up any cameras with overlapping FoV with itself. Thus, it is an interesting study to investigate whether deploying motion sensors and cameras together or apart provides overall better performance.

For the detached scenario, whenever the motion sensor is triggered, it will broadcast a long preamble with length T_p, e.g., as described in the SpeckMAC protocol [151]. The long preamble consists of repetitive blocks of the motion sensor's own ID, which are of length T_t. On the receiving camera side, each receiver has a pre-loaded list, which contains the ID numbers of all motion sensors that can trigger itself. This list can be formed during the set-up phase of the system, and used throughout the system lifetime. In order to save energy, transceivers on cameras are duty-cycled. Each receiver wakes up for T_l seconds every T_p seconds to listen to the broadcast message over the channel. If the ID of the motion sensor falls into the receiver's caller list, the receiver will trigger the attached camera to

switch to the working state. We set $T_l = 2 \times T_t$, thus these asynchronous receivers can receive at least one entire broadcast data block in case of a transmission.

6.3 Computation of Coverage Percentage and Energy Consumption

In this section, we show how to calculate coverage and energy for both the detached and attached scenarios, as well as the hybrid scenario.

6.3.1 Detached motion sensors and cameras

We first study the scenario where motion sensors and cameras are detached.

Coverage percentage

For both motion sensors and cameras on the edges of a plane, there are three geometric placement parameters: x-axis value x, y-axis value y, and orientation α. Thus, we can build matrices Pl_{ms} and Pl_{ca} to show the placement of M motion sensors and C cameras. Pl_{ms} is an $M \times 3$ matrix, where the i^{th} row represents the placement of the i^{th} motion sensor: $[x_{m_i}, y_{m_i}, \alpha_{m_i}]$. Similarly, Pl_{ca} is a $C \times 3$ matrix, with the j^{th} row: $\left[x_{c_j}, y_{c_j}, \alpha_{c_j}\right]$.

To compute the coverage percentage, instead of complex area calculations of numerous sector intersections, we use a grid of points that lays over the area to algorithmically find the percentage of the covered area. Assume there are P equally spaced points on the flat plane, the locations of which are represented by a $P \times 2$ $Point$ matrix, where the k^{th} row denotes the location of the k^{th} point: $[x_{p_k}, y_{p_k}]$.

From matrices Pl_{ms} and $Point$, we can determine whether a specific grid point is covered by a motion sensor. Thus, we construct an $M \times P$ matrix Cov_{MP}, where

$Cov_{MP}(i,k) = 1$ if grid point k, p_k, is covered by motion sensor i, which means that the following range and angle conditions are met:

$$\begin{aligned} &(x_{p_k} - x_{m_i})^2 + (y_{p_k} - y_{m_i})^2 \leq R_m^2; \\ &\alpha_{m_i} - \tfrac{\theta_m}{2} \leq \arctan\left(\tfrac{y_{p_k} - y_{m_i}}{x_{p_k} - x_{m_i}}\right) \leq \alpha_{m_i} + \tfrac{\theta_m}{2}. \end{aligned} \tag{6.1}$$

Otherwise, i.e., if p_k is not covered by motion sensor i, $Cov_{MP}(i,k) = 0$. Similarly, we build a $C \times P$ matrix Cov_{CP}, where $Cov_{CP}(j,k) = 1$ if p_k is covered by camera j, c_j, and $Cov_{CP}(j,k) = 0$, otherwise.

From matrices Cov_{MP} and Cov_{CP}, we can derive the coverage percentage for the detached scenario COV_{det}. For a grid point p_k, if it falls into a motion sensor m_i's detection region, all cameras that have overlapped areas with m_i will be triggered, since the motion sensor does not know the exact location of the trigger event. Thus, grid point p_k is *covered*, if it is also covered by at least one camera. Therefore, COV_{det} is

$$COV_{det} = \frac{1}{P} \cdot \sum_{k=1}^{P} U\left\{\sum_{i=1}^{M} Cov_{MP}(i,k) \cdot \sum_{j=1}^{C} Cov_{CP}(j,k)\right\} \tag{6.2}$$

where $U\{\cdot\}$ is the unit step function: $U = 1$ for all variable values greater than or equal to 0, and $U = 0$ otherwise.

Energy consumption

In this chapter, we consider the total energy consumption of motion sensors, cameras, and motes. The energy spent on video transmission is not considered, since the transmission energy is highly related to transmission distance, video quality and other factors. Since a single grid point can trigger one or more motion sensors, and every triggered motion sensor can further trigger one or more cameras, we need to build an $M \times C$ trigger matrix Tr_{MC} with entries

$$tr_{i,j} = U\left\{\sum_{k=1}^{P} \left(Cov_{MP}(i,k) \cdot Cov_{CP}(j,k)\right)\right\}, \tag{6.3}$$

where $tr_{i,j} = 1$ if motion sensor i can trigger camera j, and it is zero if motion sensor i cannot trigger camera j.

Therefore, for each grid point k, the number of cameras that it triggers is

$$N_{ca}(k) = \sum_{j=1}^{C} \left(U \left\{ \sum_{i=1}^{M} \left(Tr_{MC}(i,j) \cdot Cov_{MP}(i,k) \right) \right\} \right). \tag{6.4}$$

To calculate the energy on preamble transmissions, we also need to know the number of motion sensors triggered by point k, which is

$$N_{ms}(k) = \sum_{i=1}^{M} Cov_{MP}(i,k). \tag{6.5}$$

We calculate all energy components per day, and list their expressions in (6.6). We define two daily activity levels: low and high. The number of trigger events per day N_{act} is 1 and 45 for low and high activity level, respectively. We assume that motion sensors with sensing power consumption P_{ms} are on for the entire day. Each camera's power is P_{ca}, and operates for T_{on} when triggered, and automatically shuts down afterwards. E_t is the energy for long preamble broadcasting for each motion sensor per day. E_l is the energy consumed by a camera receiver's duty-cycled listening, and transceivers consume very little energy E_s during the standby mode. Hence,

$$\begin{aligned} &E_{ms} = P_{ms} \times T_{day}; \quad E_{ca} = P_{ca} \times T_{on} \times N_{act}; \\ &E_t = P_t \times T_p \times N_{act}; E_l = P_l \times T_l \times T_{day}/T_p; \\ &E_s = P_s \times T_{day}. \end{aligned} \tag{6.6}$$

The total energy consumption is calculated as

$$\begin{aligned} E_{tot} =& \frac{1}{P} \cdot \sum_{k=1}^{P} \{ E_{ca} \times N_{ca}(k) + E_t \times N_{ms}(k) \} \\ &+ (E_l + E_s) \times C + (E_{ms} + E_s) \times M. \end{aligned} \tag{6.7}$$

6.3.2 Attached motion sensors and cameras

In the attached scenario, cameras are fully controlled by the connected motion sensors, and can be only triggered by events within the motion sensor's detection range, though the camera may have a longer capture range. Also, we assume motion sensors only connect to cameras that overlap with the motion sensor's FoV. Thus, motion sensors will not trigger other co-located but not overlapped cameras unnecessarily.

Coverage percentage

A grid point is covered if it is covered by at least one attached motion sensor and camera pair. From matrices Pl_{ms} and Pl_{ca}, we derive an $M \times C$ attachment matrix Att_{MC}, where entries with value 1 mean motion sensors and cameras are attached, and values 0 mean detached. Thus, the coverage percentage is

$$COV_{att} = \frac{1}{P} \cdot \sum_{k=1}^{P} U\{\sum_{i=1}^{M}\sum_{j=1}^{C}(Cov_{MP}(i,k) \cdot Cov_{CP}(j,k) \cdot Att_{MC}(i,j))\}. \tag{6.8}$$

Energy consumption

The trigger matrix $Tr^{\dagger}_{MC}$ of the attached scenario has entries: $tr^{\dagger}_{i,j} = 1$ when motion sensor i and camera j are co-located, and also camera j's center line lies in motion sensor i's FoV. Thus,

$$tr^{\dagger}(i,j) = Att_{MC}(i,j) \cdot \left(1 - U\left\{\left|\alpha_{m_i} - \alpha_{c_j}\right| - \frac{\theta_m}{2}\right\}\right). \tag{6.9}$$

The number of triggered cameras for point p_k is then

$$N^{\dagger}_{ca}(k) = \sum_{j=1}^{C}\left(U\left\{\sum_{i=1}^{M}\left(Tr^{\dagger}_{MC}(i,j) \cdot Cov_{MP}(i,k)\right)\right\}\right). \tag{6.10}$$

Since the attached scenario does not use transceivers for communication, the total energy consumption is just the sum of the energy consumed by motion sensors and cameras, i.e.,

$$E^{\dagger}{}_{tot} = E_{ms} \times M + \frac{1}{P} \cdot \sum_{k=1}^{P} E_{ca} \times N^{\dagger}{}_{ca}\,(k). \tag{6.11}$$

6.3.3 Hybrid scenario

As an alternative approach, we introduce more flexibility into the placement design by allowing detached and attached cameras to coexist. We do not show the formulation of this hybrid approach in this chapter, but we provide the performance results in Section 6.5.

6.4 Three Different Optimization Objectives

In our study, for the detached, attached, and hybrid scenarios, our goal is to find the optimal motion sensor and camera placement strategies, to meet three different objectives: minimum energy consumption, maximum system lifetime, or lowest monetary cost for the system.

6.4.1 Objective 1: Minimize energy consumption

The first optimization problem's objective is to design a monitoring system that consumes the least energy, while meeting a certain coverage percentage threshold COV_{th}. This problem can be written as

$$\begin{aligned} \min_{\{Pl_{ms}, Pl_{ca}\}} \quad & E_{tot} \\ s.t. \quad & COV \geq COV_{th}, \end{aligned} \tag{6.12}$$

where $COV = COV_{det}$ for the detached scenario, $COV = COV_{att}$ for the attached scenario, and $COV = COV_{hyb}$ for the hybrid scenario, respectively.

6.4.2 Objective 2: Maximize network lifetime

We define the network lifetime as the lifetime of the system component that depletes its battery first. To numerically evaluate the effect of communication on the energy consumption and lifetime, we take motes [152] as the communication devices. Since cameras consume much more power than motion sensors and motes, their short lifetime is the bottleneck. To maximize the system lifetime, we aim to maximize the lifetime of the camera with the shortest lifetime.

Let $N_{pt}(j)$ define the number of points that can trigger camera j. Then,

$$N_{pt}(j) = \sum_{k=1}^{P} U\left\{\sum_{i=1}^{M}\left(Cov_{MP}(i,k)\cdot Tr_{MC}(i,j)\right)\right\}. \tag{6.13}$$

Note that if a grid point is not within the camera's FoV, but in a motion sensor's FoV, the grid point can also trigger the camera. Thus, we cannot simply use the number of grid points covered by a camera as the number of points that can trigger the camera.

For the detached scenario, we assume that the connected motes and cameras share the same battery. The battery capacity consumed by the camera motes' duty-cycled listening for one day is $C_l = I_l \times 24 \times T_l/T_p$ in milliampere per hour (mAh), the battery capacity consumed by the camera motes' standby mode for one day is: $C_s = I_s \times 24$, and the battery capacity consumed by camera j for one day is $C_{ca}(j) = I_{ca} \times \frac{T_{on}}{T_{day}} \times 24 \times \frac{N_{pt}(j)}{P} \times N_{act}$, where $N_{pt}(j)$ is the number of points that can trigger camera j, and their ratio means the probability that camera j will be triggered, if we assume that the activity is uniformly distributed in the room. Let C_{tot} denote the total battery capacity. Then, for the detached scenario, the lifetime of this single camera is

$$L_{det}(j) = \frac{C_{tot}}{C_l + C_s + C_{ca}(j)}. \tag{6.14}$$

The lifetime of the attached scenario is:

$$L_{att}(j) = \begin{cases} \frac{C_{tot}}{C_{ca}(j)}, N_{act} \neq 0 \\ L_{ms}, N_{act} = 0, \end{cases} \tag{6.15}$$

where L_{ms} is the lifetime for motion sensors.

The constraints are coverage and cost of devices. Hence, the optimization model can be summarized as

$$\begin{aligned} \max_{\{Pl_{ms}, Pl_{ca}\}} \quad & \arg\min_{C} L(j) \\ s.t. \quad & COV \geq COV_{th} \\ & K \leq K_{th} \end{aligned} \tag{6.16}$$

where K denotes the costs, and K_{th} is the budget.

6.4.3 Objective 3: Minimize monetary cost

Since some users may want to minimize the total monetary cost of the system, another optimization problem is to minimize the total cost, with constraints on coverage and network lifetime. Thus, we can summarize this problem as

$$\begin{aligned} \min_{\{Pl_{ms}, Pl_{ca}\}} \quad & K \\ s.t. \quad & COV \geq COV_{th} \\ & \arg\min_{C} L(j) \geq L_{th} \end{aligned} \tag{6.17}$$

where L_{th} denotes the network lifetime threshold.

6.5 Numerical Results and Analysis

6.5.1 Devices and parameters used

The devices we use to build wireless video monitoring systems are listed in Table 6.1. Cameras are set to be working for 1 minutes when triggered, i.e., $T_{on} =$

Table 6.1: Devices used for wireless video monitoring systems.

Devices	**Motion Sensor**	**Mote**	**Camera**
Details	PIR motion sensor [153]	Tmote Sky mote [152]	wireless camera [154]
Range (m)	3	50	5
Voltage (V)	2.4-3.6	2.1-3.6	9
Current (mA)	0.01	TX: 19.5, RX: 21.8	71.1
Power (mW)	0.03	TX: 58.5, RX: 65.4	640

$60s$, the preamble transmission lasts for $T_p = 10s$, duty-cycled camera motes wake up for $T_l = 2 \times T_t = 60ms$ for every T_p, and one entire day is $T_{day} = 86400s$.

We consider a room with size 5 m x 3 m. Neighboring grid points are 0.5 m apart. Also, the possible placement positions of motion sensors and cameras can only be on integer points along the edges of the 1.5m-high flat plane illustrated in Fig. 6.1. Since the specification of the motion sensors we use define a 90-degree angle of FoV [153], we assume that they adopt discrete orientations, with the center line deviating from the edge of the plane for 45, 90, or 135 degrees. Since the cameras' angle of FoV is 38 degrees [154], the possible orientations for cameras are: 19, 55, 90, 125, or 161 degrees.

For the detached scenario, each motion sensor and the connected mote costs \$20, and each camera and the connected mote costs \$60. For the attached scenario, each motion sensor costs \$15 [153], and each camera costs \$60 [154].

6.5.2 Simulation results

Using MATLAB, we implement the formulas given in Sections 6.3 and 6.4, and find the optimal motion sensor and camera combinations for our objectives. The simulation results for Objective 1 are given in Fig. 6.2, which shows the minimum energy consumption for the detached scenarios with different numbers of motion sensors and cameras at low activity level. It is shown that to achieve 90% coverage,

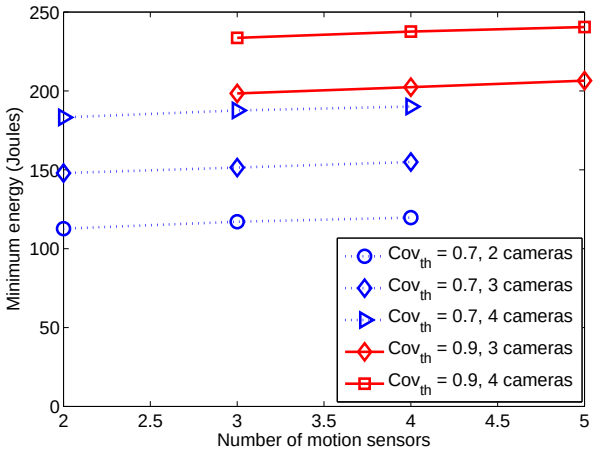


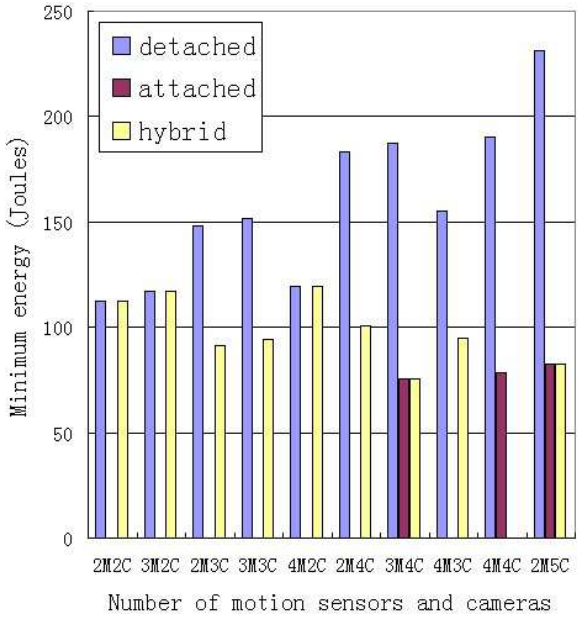
Figure 6.2: The minimum total energy consumption for the detached scenario at low activity level under coverage constraints of 70% and 90%.

higher energy is needed. Thus there is a tradeoff between energy and coverage. Also, as the number of cameras increases, the energy increases as well. For low activity level, the energy for cameras are comparable with the motes' duty-cycled listening energy. Thus, the difference in energy between 70% and 90% coverage constraints is not large. We also run the simulation for high activity level, and we find that cameras consume much higher energy, and the difference between 70% and 90% coverage constraints is large.

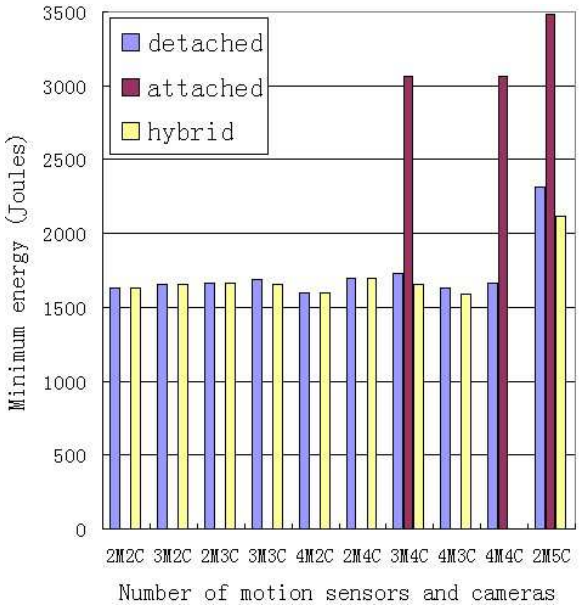
Fig. 6.3 shows the minimum energy consumption for the detached, attached, and hybrid scenarios at low and high activity levels. For a small number of motion sensors and cameras, we do not have data for the attached scenario, since it cannot provide the required coverage. We can see that the hybrid approach always consumes less than or equal to the energy of the other two approaches, since it has more flexibility in placement strategies. We also observe that, at low activity level, the attached approach consumes less energy than the detached approach, but at high activity level, the detached approach is better. This is because the detached approach consumes extra energy on motes, the amount of

which is comparable with the energy consumed by cameras at low activity level, but is relatively low at high activity level. Thus, the detached approach performs worse than the attached scenario at low activity level. Motes help increase the range of cameras, but the duty-cycled listening consumes more energy, thus a tradeoff exists.

To obtain the optimal placement strategy for Objectives 2 and 3, we use the cost and lifetime coordinates as shown in Fig. 6.4 for high activity level. Each dot represents the longest lifetime for the corresponding amount of cost of devices. This figure shows that, in general, higher cost of devices gives longer lifetime, thus there is a tradeoff between lifetime and cost of devices. We can see that at high activity level, our monitoring system can last more than 3 months. The detached scenario generally costs less than the attached scenario, since it requires fewer cameras to obtain the same coverage. This is because the cameras' FoV in the attached scenario is limited by the motion sensors' FoV, and thus each camera can only cover a limited area. For Objective 2, i.e., the maximum lifetime, a line can be drawn at the cost threshold, and for all dots below the line, the right most dot represents the case with the maximum lifetime. For example, if the user's budget is \$300, and the coverage requirement is 70%, the dot pointed by the arrow is the optimal solution. We plot the placement scheme for this optimal case in Fig. 6.5, which uses 4 cameras detached from 2 motion sensors. The camera's coverage is represented by a horizontal pattern, and areas with cross pattern are covered by both motion sensors and cameras. The cameras' coverage is slightly overlapped, as shown in the bright yellow stripe areas. We can see from Fig. 6.5 that although the motion sensors and cameras are co-located, they are detached, since they have different orientations. Similarly, to get the optimal placement for Objective 3, i.e., the minimum monetary cost, we can draw a vertical line at the lifetime constraint, and then find the lowest point on the right side of the line, which gives the lowest cost of devices.



(a)



(b)

Figure 6.3: The minimum total energy consumption for different scenarios at a) low and b) high activity level under coverage constraints of 70%.

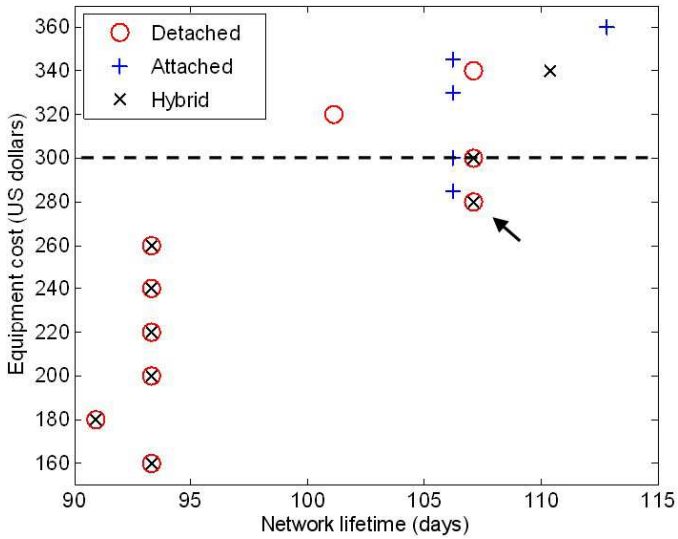


Figure 6.4: Cost of devices vs. network lifetime at high activity level, under coverage constraint of 70%.

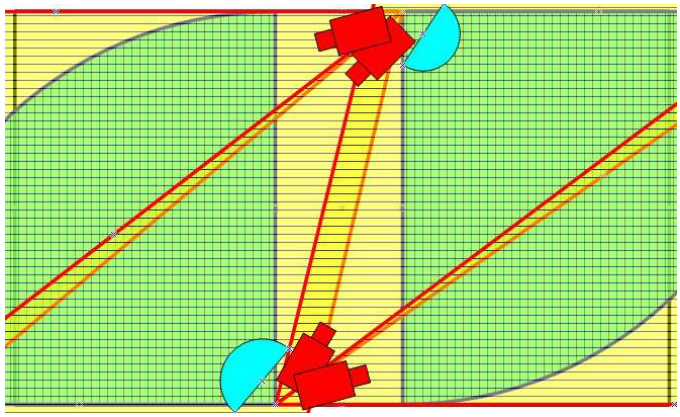


Figure 6.5: Optimal placement scheme (detached case) to achieve the maximum network lifetime, under coverage constraint of 70% and cost constraint of $300.

We also run simulations for lifetime per cost and lifetime per battery for different numbers of motion sensors and cameras at high activity level. Results show that 2 cameras detached from 2 motion sensors achieves longest lifetime per cost and longest lifetime per battery.

6.6 Conclusions

A motion sensor and camera deployment optimization is presented in this chapter, and issues of coverage, energy consumption, network lifetime, and cost of devices are investigated. Simulation results show that to meet the same coverage constraint, compared with the detached scenario, the attached scenario consumes less energy at low activity level, but more energy at high activity level. A hybrid approach is also proposed, which can use both detached and attached cameras. Due to its advantage in flexibility, the hybrid approach always achieves better or the same performance. For a certain coverage and cost constraint, an optimal placement strategy is given to achieve the maximum network lifetime. We also examine the impact of cost, battery and activity level on placement strategies.

Part III

Application of Sensor Network Coverage Algorithms to Protein Structure Prediction

7 Algorithms for Protein Structure Prediction

7.1 Introduction

The general problems of detection, estimation and prediction have been important research subjects to both the structural bioinformatics and communications research communities. Over the years, successful theories and technologies have been developed in both fields to tackle these problems. However, collaborations between the two fields rarely occur, mostly due to the seemingly significant differences in their research subjects. We aim to exploit the potential linkage between structural bioinformatics and communications through the application of well-developed algorithms for sensor network placement to the prediction of protein structures, a difficult challenge in structural bioinformatics and a significant bottleneck in pharmaceutical and biomedical research.

In the pharmaceutical and biotechnology industries, protein structure prediction is important for many applications, such as novel enzyme design and antibody therapeutics. Accurate side chain prediction is key to modeling protein-protein interactions and predicting the effects of mutation. Also, protein structure prediction is the basis for structure-based drug design and side-effect studies. Rational drug design has been widely used within the pharmaceutical and biotechnology

industries to deliver innovative healthcare solutions to patients. A prerequisite of such a design process is the understanding of the atomic structures of the target protein and the drug molecule. However, experimental derivation of the atomic structures is both time-consuming and costly. Thus, computational prediction of protein structures at atomic resolution is not only an interesting academic challenge, but also a powerful tool that can be readily applied to pharmaceutical research and impact the lives of millions of patients.

A major difficulty in protein structure prediction is the excessive computational complexity caused by an extremely large sampling space. The number of optimization parameters and the number of constraints needed to determine the optimum protein structure grow exponentially with the number of residues in a protein. A small group of residues each with a few rotation degrees of freedom already yields a huge number of possible side chain conformations. Therefore, an efficient sampling strategy is highly desired to explore the large search space, such that a near native protein structure can be found in a reasonable amount of time with high accuracy and confidence.

In this study, we explore the application of sensor placement optimization strategies developed for wireless sensor networks (WSNs) applied to protein side chain prediction, considering both the inter-residue and intra-residue aspects of the side chain conformation scoring function and avoiding exhaustive search by driving the rotamer sampling towards the direction that favors the potential energies. The covered sensing areas in WSNs are represented by the three dimensional space occupied by atoms both on the backbone and on side chains of proteins. Algorithms proposed for sensor placement optimization are adapted to predict the conformations of protein side chains. Our preliminary benchmark results show that the proposed algorithm can effectively reduce the number of atom collisions compared to an initialized predicted structure.

The rest of the chapter is organized as follows: Section 7.2 describes the side

chain prediction problem. Section 7.3 presents state-of-the-art algorithms for side chain prediction that were proposed in bioinformatics. The similarities between the side chain prediction problem and the concepts used in the WSN coverage problem are provided in Section 7.4. Section 7.5 describes the proposed side chain prediction algorithm, and the performance evaluations are presented in Section 7.6. Section 7.7 concludes this chapter.

7.2 Side Chain Prediction Background

Proteins are formed by chains of amino acids joined together by peptide bonds, resulting in chains of residues. The protein structure consists of a polypeptide backbone and side chains. The amino acids sequence and the backbone dihedral angles (ϕ, ψ) specify the conformation of the backbone. There are 20 different types of amino acids, each having up to 4 dihedral angles, denoted as χ_1 to χ_4. For the side chain prediction problem, the backbone structure is assumed to be given, and thus only the conformations of side chains are to be determined. Rotations around χ angles create an immense number of possible structural forms, even with a stationary backbone.

An example of the backbone dihedral angles (ϕ, ψ) and side chain χ angles for residue Asparagine (ASN) is provided in Fig. 7.1. The three-dimensional (3D) model of the protein structure is viewed using the software Chimera [9]. The backbone is formed by repetitions of 'N-CA-C-O' atoms, and side chains are clusters of atoms connected to the CA atom. Backbone dihedral angle ϕ is determined by 'C (on the previous amino acid)-N-CA-C', and ψ is determined by 'N-CA-C-N (on the next amino acid)'. For residue ASN, side chain dihedral angle χ_1 is determined by 'N-CA-CB-CG', and χ_2 is determined by 'CA-CB-CG-OD1'.

The side chain conformation of a residue is uniquely determined by its torsion

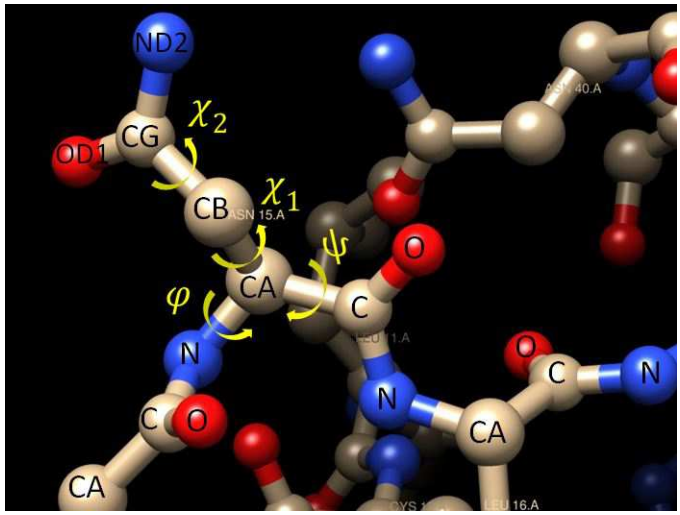


Figure 7.1: Backbone and side chain dihedral angles for the 15^{th} residue ASN on protein 1Q9Ba, visualized using Chimera [9].

angles. Plausible conformations of a side chain can be clustered into a limited number of discrete states, named 'rotamers'. Thus, side chain prediction is equivalent to finding the optimal combination of the rotamers that stabilize the protein structure. Structural bioinformatics and other statistical studies have been conducted to analyze the frequencies of rotamer occurrences based on a large set of proteins with known structures. The Backbone Dependent Rotamer Library (BBDRL) [10] from Dunbrack's lab has been widely used as a key resource for protein structure prediction. The BBDRL consists of rotamer frequencies, as well as the mean and variance of side chain dihedral angles, i.e., χ angles. For each individual type of side chain, the rotamer probabilities are given on a $10° \times 10°$ grid of the local backbone dihedral angles (ϕ, ψ), regardless of the secondary structure of the protein.

An example of the BBDRL is shown in Fig. 7.2 for residue Leucine (LEU). Columns 2 and 3 denote the backbone dihedral angles ϕ and ψ, respectively. Columns 5-8 are the bin indices for χ_1-χ_4. For most types of side chains, each χ angle can be represented by one of three bins. Some residues can have up to

LEU	20	-100	0	1	1	0	0	0.000000	61.1	80.5	0.0	0.0
LEU	20	-100	0	1	3	0	0	0.000000	74.6	-56.6	0.0	0.0
LEU	20	-90	0	3	2	0	0	0.776377	-66.2	173.8	0.0	0.0
LEU	20	-90	0	2	1	0	0	0.132654	-179.7	62.1	0.0	0.0
LEU	20	-90	0	2	2	0	0	0.081027	-171.2	161.4	0.0	0.0
LEU	20	-90	0	3	1	0	0	0.009310	-81.8	64.0	0.0	0.0
LEU	20	-90	0	2	3	0	0	0.000608	-173.7	-72.2	0.0	0.0
LEU	20	-90	0	3	3	0	0	0.000017	-80.6	-60.7	0.0	0.0
LEU	20	-90	0	1	2	0	0	0.000006	70.9	166.6	0.0	0.0
LEU	20	-90	0	1	1	0	0	0.000000	61.1	80.5	0.0	0.0
LEU	20	-90	0	1	3	0	0	0.000000	74.6	-56.6	0.0	0.0
LEU	20	-80	0	3	2	0	0	0.660353	-66.1	173.8	0.0	0.0
LEU	20	-80	0	2	1	0	0	0.199488	-179.7	62.0	0.0	0.0

Figure 7.2: Rotamer probabilities for residue LEU in the backbone dependent rotamer library (BBDRL) [10].

12 bins for each χ angle. We can see that both χ angles in residue LEU have three bins. Therefore, there are $3 \times 3 = 9$ combinations in total for the side chain conformation of LEU. Column 9 is the rotamer probability, and columns 10-13 are the mean values for χ_1-χ_4. The variance values for χ_1-χ_4 are not shown in the screenshot.

For the rotamer highlighted in Fig. 7.2, if the dihedral angles for the rigid backbone are known to be $(\phi, \psi) = (20°, -90°)$, the mean χ values for the most preferred rotamer, i.e., the rotamer with the highest probability, are $\chi_1 = -66.2°$, and $\chi_2 = 173.8°$. The probability for this most preferred rotamer is 77.6%.

The rotamer preferences in BBDRL [10] were determined by studying 3,985 protein chains, which contain 581,128 residues. There are nearly 70,000 experimentally determined protein structures deposited in the Protein Data Bank (PDB) [155] as of today. These known protein structures have proven to be an invaluable asset to obtain the statistical structural information. Around 90% of the protein structures available in the PDB were determined by X-ray crystallography, and were submitted by biologists and biochemists from around the world. In this study, we evaluate the performance of our proposed side chain prediction algorithm on these known protein structures determined by X-ray crystallography.

7.3 Related Work

A widely-adopted approach for side chain prediction is to model a potential energy function of the protein, and then search for the rotamer confirmations with the lowest energy score. The scoring function is usually based on two types of information: intra-residue potential and inter-residue potential. The intra-residue potential reflects the inherent conformational preference of a given residue, which is modeled by using the rotamer probabilities in the library: the higher the rotamer probability, the lower the intra-residue potential is. The inter-residue potential models an abstract interaction resulting from the joint effect of all atom-level interactions between the residues in consideration and all other neighboring residues in the 3D space.

Both potential energies can be derived from the statistical structural information. Due to the large number of residues of a protein and the numerous possible rotamer combinations for each side chain, it has been shown that for energy functions typically used in side chain prediction, finding the global optimum is NP complete [156]. To reduce the computational complexity of this combinatorial problem, dead-end elimination [157] [158] and self-consistent mean field [159] methods have been proposed.

Software has been developed for side chain prediction as well, such as the widely-used SCWRL4 [160]. In SCWRL4, the self energy for one side chain is modeled by rotamer probabilities, and the pair energy is modeled by soft van der Waals atom interaction potential. The search strategy of SCWRL4 is based on the dead-end elimination method. In addition, a tree decomposition algorithm is used to solve the combinatorial packing problem.

The major difference between our proposed algorithm and the aforementioned algorithms is the way that we model inter-residue interactions. Classical approaches study the interactions between atom pairs through chemical or physical

properties such as van der Waals forces, while we consider the atom collisions and atom spatial density from a geometry perspective. Although this concept was inspired by theories from communication research, this study is still based on the fundamentals of biology structure, such as the rotamer statistical preference.

7.4 Directional Sensor Network Deployment vs. Side Chain Prediction

The similarities between the sensor placement optimization problem and the side chain prediction problem are the foundation of this interdisciplinary work. Wireless sensor networks (WSNs) have been widely used for emergency disaster response, event detection, and object tracking. Directional sensor networks (DSNs) are one type of WSN, where each directional sensor node covers a certain shape of area, named the Field of View (FoV). For example, the FoV for a directional camera is the surveillance area. DSNs may be deployed in harsh environments such as rain forests or deserts, where deployment using manpower is difficult. Under such conditions, sensor nodes may be initially deployed randomly or at a limited precision in the field by helicopters or robots.

The coverage problem is essential for DSNs. Take the surveillance network as an example, a certain coverage percentage should be achieved to prevent misdetections of any events of interest [22] [161]. On the other hand, if two sensors are placed too close to each other, their FoVs may be greatly overlapped. Thus, it is very likely that the two sensors are triggered redundantly, resulting in a waste of resources (e.g., energy). Note that reducing energy usage of the sensor is crucial in order to maximize the lifetime of the entire WSN.

To optimize the sensor placement in DSNs, sensors may adjust their FoVs by changing their sensing orientations or changing their locations after the initial

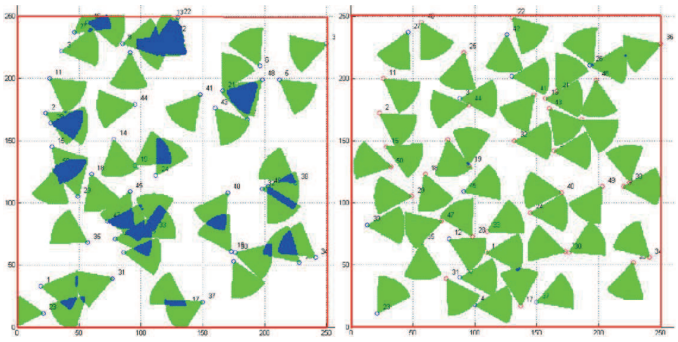


Figure 7.3: Directional sensor network coverage before and after sensor placement optimization. Reprint from [11].

deployment. For example, sensors equipped with motors and some limited energy may flip themselves to a nearby location. Through self-adjustments and communication between each other, sensors coordinate and optimize their locations.

For large-scale DSNs, obtaining an optimal or near optimal sensor placement is a computationally intensive process. Hence, a number of movement strategies have been proposed for DSNs to achieve a certain percentage of coverage, while maintaining a minimum numbers of sensor movements [11] [162] [45].

Fig. 7.3 shows an example of DSN coverage before and after sensor placement optimization (reprint from [11]). The sensors' FoVs are represented by green pie-shaped areas, and overlapped FoVs are denoted by blue areas.

To map the sensor placement problem in DSNs to the protein side chain prediction problem, we model the space occupation of an atom as a directional sensor's FoV. Atom packing and side chain rotation can be modeled as sensor placement and change of sensing orientation. The goal of minimizing coverage overlap in DSNs is similar for atoms in a native protein structure, where the existing repulsive force prevents two atoms from being too close to each other. Similar to the goal of minimizing coverage holes in DSNs, a native protein structure usually has no large voids inside its hydrophobic core, in order to maintain the stability of the entire structure.

Table 7.1: Similarities between DSN deployment and protein side chain prediction.

	DSN	Side chain
Optimization goals	minimize coverage overlap	minimize atom collisions
	minimize coverage holes	minimize voids
Optimization constraints	limited sensor movement	restricted spatial occupation
	energy cost for changing sensor orientation or position	energy cost for rotating rotamer away from favorable position

There are two constraints for both the aforementioned problems. While sensor movement is limited to the local area, side chain spatial occupation is restricted to the connection with the backbone. Also, similar to consuming energy when moving sensors away from their initial locations, we introduce cost in the energy score function when tuning rotamers away from their most preferred angles. These common optimization goals and constraints are the bases for us to use algorithms developed for the sensor placement problem to solve the side chain prediction problem. We summarize the similarities between the two problems in Table 7.1.

7.5 Proposed Side Chain Prediction Algorithm

We propose a novel greedy algorithm for the side chain prediction problem. A greedy algorithm is an algorithm that solves the problem using a heuristic of making a locally optimal choice at each stage, with the goal of approximating a global optimal solution in a reasonable time. A flowchart of our proposed side chain prediction algorithm is shown in Fig. 7.4. Each step of the algorithm is explained as follows.

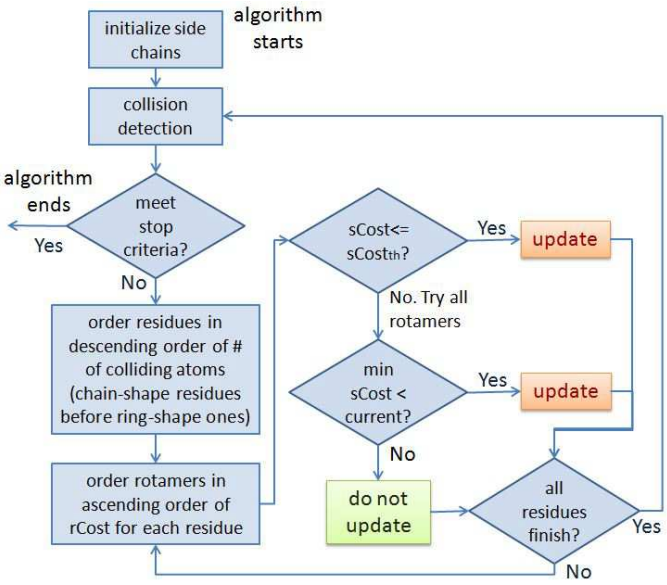


Figure 7.4: The proposed algorithm for protein side chain prediction.

7.5.1 Step 1: Initialize side chain structure

In order to maintain a low intra-residue potential, we initialize all side chains with their most preferred χ angles, i.e., rotamers with the highest probabilities in the rotamer library. For some initialized structures, side chains collide with the backbone. Since the backbone is pre-determined, and no collisions should exist in native protein structures, we do not consider these rotamer options in the side chain prediction problem. In this case, the rotamers with the second highest probabilities are chosen instead. This derived side chain structure is used as the starting point for the greedy algorithm for further optimization.

7.5.2 Step 2: Detect atom collisions

Collision detection using weighted space

The initialized structure derived from Step 1 results in many atom collisions, since the rotamer library does not consider interactions between two residues,

but only considers one residue at a time. To detect atom collisions, measuring the distance between every atom pair is computationally intensive. Instead, we build a 'weighted space' where different weight values are given to space units to represent the atom occupancy on that unit. Therefore, the sum over a volume in the weighted space is a direct indicator of the space density for that certain area, and we can easily obtain a global knowledge of the dense area and the sparse area in the 3D space. The space density information is the basis for us to optimize atom placement.

As shown in Fig. 7.5, we model an atom with a sphere with different weight values in the inner part and outer part. The weight values for space units in the inner sphere are set to be w_i, and the weight values for space units in the outer sphere are set to be w_o. The weight for a space unit is the sum of the weight values of all the covering atoms on that unit. The radii for the inner sphere and outer sphere are r and R, respectively. R is the radius of the atoms, which differs by elements and residues. Since favorable interactions can be formed between atoms, such as hydrogen bonds, an overlap with distance d_v is allowed between two contacting atoms. Thus, the threshold distance between the centers of two contacted atoms with radii R_1 and R_2 is $d_{th} = R_1 + R_2 - d_v$. If the distance between these two atoms is smaller than d_{th}, these two atoms are considered as colliding.

We set the radius for the inner sphere r to be $r = R - d_v$. Fig. 7.5 shows the scenario where two atoms are on the edge of colliding. If these two atoms are placed closer, the outer sphere of one atom will overlap with the inner sphere of the other atom, resulting in space units with weights greater than w_i. Therefore, space units with weights greater than w_i are indications of atom collisions.

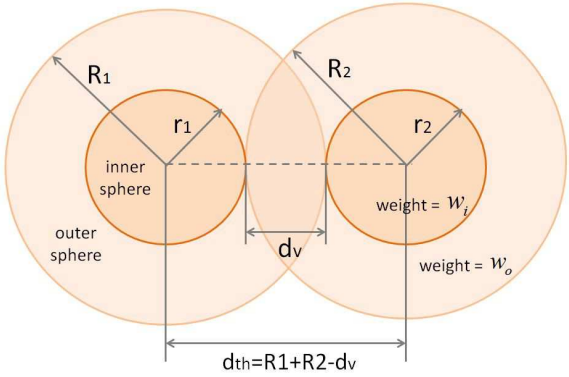


Figure 7.5: Weighted space for two atoms at the edge of colliding.

Recreate protein structure in weighted space

For two atoms connected by covalent chemical bonds, the distance between them is much smaller than the distance threshold d_{th}. If we add atoms on the entire protein structure to the weighted space all together, there will be quite a number of space units with weight greater than w_i but are not caused by collisions. Therefore, we recreate the protein structure in the weighted space part by part.

First, we add all the backbone atoms to the weighted space. Since all the atoms on the backbone are connected by chemical bonds, and there are no collisions within the backbone, we can safely 'smooth out' the backbone by changing all space units with weights greater than w_i to w_i. Secondly, we assemble atoms within each individual side chain, with the rotamers set to be their most preferred χ angles. Similarly, since all the atoms within one residue are bonded, and we assume that there are no collisions within the side chain for conformations chosen from the rotamer library, we can smooth out each individual side chain by changing all space units with weights greater than w_i to w_i. The last step is to assemble all the side chains to the backbone by adding all the side chains to the weighted space, and smoothing out the joint region between the CA atom on the backbone and the CB atom on the side chain.

For this recreated structure in the weighted space, the space unit weights for

bonded atoms and atoms that are in contact but not collided have weights smaller than or equal to w_i, and the space units for collided atoms have weights greater than w_i. With the knowledge of the position of the collisions as well as the space density information, we determine the process to solve collisions in the next step.

7.5.3 Step 3: Solve atom collisions

Determine residues to move

We solve atom collisions by rotating residues, but this might cause additional collisions. We sort residues in descending order of the number of colliding atoms, and start to solve collisions from the residues with the largest number of colliding atoms. When counting the number of colliding atoms for one residue, we only count the colliding atoms on the residues except the CB atom, since the backbone atoms and the CB atom cannot be moved.

Rotating ring-shape residues results in the movement of a large number of atoms, which means a higher risk of causing additional collisions. Therefore, we solve collisions for chain-shape residues first, and then solve collisions for ring-shape residues. Ring-shape amino acids include: Histidine (HIS), Phenylalanine (PHE), Proline (PRO), Tryptophan (TRP), and Tyrosine (TYR).

Determine χ angles to move

To update a residue with colliding atoms, we retrieve all potential rotamers from the rotamer library. For example in Fig. 7.2, to update a LEU residue with backbone dihedral angles (ϕ, ψ)=(20°,−90°), all corresponding rotamers are considered, i.e., line 3 to line 11. For a residue with multiple χ angles, rotamers that do not solve the current collision should be deleted. Take the residue ASN shown in Fig. 7.1 as an example, if atom CG collides with another atom, rotating χ_2 alone cannot move CG. χ_1 or both χ_1 and χ_2 have to be moved. The rotamers

that do not change χ_1 are deleted to save the computation cost. We build a lookup table with inputs being the residue name, the index of the colliding atom, and the output being a proper χ to be moved to solve the colliding atom.

Determine new χ angles

We build two cost functions to determine the new χ angles: the rotamer preference cost $rCost$, which takes into consideration the rotamer preference, and the space density cost $sCost$, which takes into consideration the density of the new location, and whether rotating to the new location will cause additional collisions. We choose the potential rotamer with the lowest costs to update.

The rotamer preference cost function $rCost$ depicts the penalty by moving rotamers away from their most preferred positions. $rCost$ is defined by the inverse of the probability for the new rotamer P_r, as presented in (7.1):

$$rCost = \frac{1}{P_r}. \tag{7.1}$$

When rotating a side chain, we prefer the new location to be a sparse area, as this movement may also fill up voids in the protein structure. To calculate the space density for a side chain, we calculate the space density for all moveable side chain atoms, i.e., starting from the next atom of CB. The space density for an atom is modeled by the sum of the weights for the units within a cube region with length L surrounding that atom. The space density for a side chain is the average of the space density for all moveable side chain atoms.

In addition to a relatively low space density, we prefer the new location to have fewer collisions than the number of collisions for the current conformation. Therefore, we modeled the space density cost $sCost$ in (7.2):

$$sCost = \alpha_1 \cdot e^{\frac{K_2 - K_1}{K_1}} + \alpha_2 \cdot e^{\frac{D_2 - D_1}{D_1}}, \tag{7.2}$$

where α_1 and α_2 are coefficients. K_1 and K_2 are the number of colliding atoms for the current structure and the new structure with the residue set to the new

rotamer, respectively. To speed up the program, we only run collision detection at a local region to calculate K_1 and K_2. D_1 and D_2 are the space density for the side chain with the current position and the new position, respectively. We use an exponential function for both terms since we allow a small amount of increase in either the number of colliding atoms or the space density, but a huge increase is not allowed. If $K_1 = K_2$, and $D_1 = D_2$, $sCost = \alpha_1 + \alpha_2$. In order to achieve a lower $sCost$ after updating each residue, $sCost$ for the new rotamer should be smaller than $\alpha_1 + \alpha_2$.

Since it is less computationally intensive to calculate the rotamer preference cost $rCost$ than the space density cost $sCost$, we first calculate $rCost$ for all rotamer options, and order the rotamer options in ascending order of $rCost$, as shown in Fig. 7.4. Then we calculate the space density cost $sCost$ for the rotamer options with the lowest $rCost$, and compare it with a pre-defined $sCost$ threshold $sCost_{th}$. If the calculated $sCost$ is smaller than or equal to $sCost_{th}$, we update the residue with this new rotamer.

If all rotamer options have been tried, but no option meets this criterion, we choose the minimum $sCost$ among all rotamer options. If this minimum $sCost$ is smaller than $\alpha_1 + \alpha_2$, we update the residue with the rotamer with the minimum $sCost$. Otherwise, we do not update the residue for this iteration, which means that all the other rotamer options have a higher space density cost than the current rotamer. The current collisions on this residue may be resolved by moving the other colliding atom, or we will try to update this residue on the next iteration, when more space may be left to choose other rotamer options.

When all residues have been updated or choose to not move, we run collision detection for the entire protein structure, and iterate Step 3 again.

7.5.4 Step 4: Algorithm ends

The proposed greedy algorithm ends when any one of the following criteria is met: 1) no collisions remain for the entire protein structure, 2) no residues are updated in the last iteration, or 3) the chosen rotamers for the last iteration are all the same with the chosen rotamers for a previous iteration, which means that the program falls into a deadlock.

7.6 Evaluation

We present the preliminary results for our proposed side chain prediction algorithm for a small sample of proteins. The side chain prediction accuracy results for our proposed algorithm are compared with SCWRL4. The numbers of colliding atoms for the initialized structure and the predicted structure are also examined.

7.6.1 Parameter settings

Parameters used in Step 2 of the algorithm are set as follows. For two contacting but not colliding atoms, the maximum allowed distance for the overlapped area is $d_v = 1.4$ Angstrom. Since the largest atom is the Oxygen atom, with a radius of 2.0 Angstrom, we assume that it can be fit into a $11 \times 11 \times 11$ units cube. Thus, the units in the weighted space are separated by $2.0 \times 2/11 = 0.36$ Angstrom. If this value is chosen to be too small, the atom positions are mapped to the weighted space with a higher resolution. However, there will be too many grids in the space, which increases the computation intensity of our proposed algorithm. Thus, we choose the unit interval value to be 0.36 Angstrom to satisfy this tradeoff. The weight value assigned for space units located in the inner sphere and outer sphere are set to be $w_i = 1$ and $w_o = 0.1$, respectively. Therefore, space units with weights greater than 1 indicate atom collisions.

Parameters used in Step 3 of the algorithm are set as follows. The length of the cube used to calculate the space density around an atom is set to be $L = 15$ units, which is slightly larger than the atom radius, i.e., 11 units. The coefficients used in the space density cost function in (7.2) are preliminarily set to be: $\alpha_1 = 1$, and $\alpha_2 = 1$. The space density cost threshold is set to be $sCost_{th} = 1.4$. In the future, we will optimize these parameters based on an evaluation of the proposed algorithm on a larger set of testing protein structures.

7.6.2 Experimental results

In this side chain prediction study, a side chain torsion angle is considered to be predicted accurately if its value is within 40° deviation from the angle values in the native side chain conformation, a prediction accuracy metric that is widely used for side chain prediction algorithms, such as [160]. For a preliminary test, we choose seven small-sized proteins from PDB to evaluate our proposed side chain prediction algorithm. As widely accepted, we report the prediction accuracy for χ_1 and the first two angles χ_{1+2}, i.e., both χ_1 and χ_2 are predicted correctly, on each of the individual tested proteins. In Table 7.2, we present the χ_1 and χ_{1+2} prediction accuracy for the predicted structures using the proposed algorithm and using the software SCWRL4 [160]. The prediction accuracy of the initialized structure derived from Step 1 of the proposed algorithm is also used as a reference. Additionally, we show the decrease of the number of colliding atoms from the initialized structure to the predicted structure in Table 7.2.

We demonstrate that the proposed algorithm can effectively reduce the number of colliding atoms for the predicted structure, which complies with the sensor coverage optimization objective in DSNs. However, the χ_1 and χ_{1+2} prediction accuracies are not improved much compared with the accuracies for the initialized structure, and are generally lower than the accuracies achieved by SCWRL4.

Table 7.2: Performance comparison for side chain prediction for our proposed algorithm and SCWRL4.

Protein	Residues	Number of colliding atoms			χ_1 accuracy (%)			χ_{1+2} accuracy (%)		
		Init.	Prop.	SCWRL4	Init.	Prop.	SCWRL4	Init.	Prop.	SCWRL4
1Q9Ba	43	2	0	0	80.56	80.56	83.33	63.89	63.89	75.00
1WUWa	45	10	0	0	68.42	65.78	76.32	52.63	52.63	60.53
1OKHa	46	8	0	0	84.21	86.84	84.21	76.32	78.95	84.21
1DTDb	61	2	0	0	75.00	71.15	76.92	59.62	55.77	69.23
1PTQ	50	12	0	0	68.18	68.18	70.45	56.82	56.82	54.55
1CTF	68	6	0	0	73.91	71.74	80.43	69.57	67.39	73.91
21BI	151	46	0	0	61.31	64.96	62.77	44.53	45.99	44.53

7.7 Conclusions and Future Work

This work translates the coverage optimization concept in directional sensor networks to the protein side chain prediction problem. We project the protein structure to a weighted space, and use both the rotamer preference and the space density to optimize the atom placement in the 3D space. Preliminary results show that our proposed algorithm can greatly reduce the collisions in the predicted structure, compared with an initialized structure, where all rotamers are set to their most preferred angles. For a testing set of seven proteins, on average, 72.6% of χ_1 angles and 59.6% of χ_{1+2} angles are predicted correctly within 40° deviation from the angle values for the native protein structures.

In this study, we optimize the atom placement in the 3D space by solving atom collisions. However, voids may exist inside the predicted protein structure. In the future, we will improve the proposed algorithm by adding one step to detect and fill up voids. Also, we will optimize the parameters used in the algorithm, and evaluate the performance of the proposed algorithm on a larger set of proteins.

8 Conclusions and Future Directions

8.1 Contributions

In this dissertation, we develop algorithms for affective sensing through emotion classification; for ubiquitous sensing through in-home video monitoring; and for protein structure prediction. The emotion classification system can be used for behavior studies such as parent-teen relationship studies; for building in-home entertainment systems; or for other applications. The video monitoring system is designed to achieve a higher coverage rate, higher energy-efficiency, longer network lifetime, and lower monetary cost. The protein structure prediction algorithm maps an approach for sensor coverage optimization to the problem of protein structure prediction, which offers information critical to pharmaceutical research, such as structure-based drug discovery and rational drug design.

The main contributions of this dissertation are summarized as follows:

- We develop a noise resilient fundamental frequency (F_0) detection algorithm named BaNa. We provide an extensive study of the performance of BaNa along with 7 other classic and state-of-the-art F_0 detection algorithms for a range of noisy speech and music signals with different types of noise and noise levels [15] [16].

- We propose a speech-based six-class emotion classification system using several one-against-all support vector machines with a threshold-based fusion mechanism to combine the individual outputs. A thorough performance evaluation of this system is provided for different test scenarios, including classifications on noisy speech data and tests using a dataset with non-professional acted emotions [17] [18] [19].

- We implement a mobile emotion sensor on a Windows Phone platform called 'Listen-n-feel' for a binary emotion classification for happy or not happy.

- We develop an image transmission model that allows the user to specify an image quality constraint. Lower layer parameters are optimized, such as transmit power and packet length, to minimize the energy dissipation in image transmission over a given distance [21].

- We explore the optimization of motion sensor and camera placement for in-home monitoring system design, in order to achieve the minimum energy consumption, longest network lifetime, or lowest monetary cost [22].

- We conduct an interdisciplinary study that combines the research fields of structural bioinformatics and communications to provide a novel solution to the problem of protein side chain prediction, which offers information critical to pharmaceutical research.

As we enter the big data era, noise is a factor that cannot be neglected. Although the algorithms proposed in the three parts of this dissertation contribute in different domains, these algorithms share the same theme of dealing with noisy data. The speech-based emotion classification system is designed to combat noise in speech; the in-home video monitoring system is designed in an energy-efficient way by allowing noisier images to be captured; and the greedy algorithm proposed for protein structure prediction deals with noisy data as well, since the x-ray crys-

tallography can only capture a snapshot of moving proteins. Hence, with the presence of noise in big data studies, we first need to identify the source of noise, and then we can design robust algorithms to combat noise, or adapt the system requirements to the amount of noise in the data.

8.2 Future Directions

The increasing demand for affective and sensing applications and for protein structure prediction motivates continued innovation. To meet this growing need, additional research is needed as follows:

We are developing an Android implementation to extract the speech features used in our speech-based emotion classification system, which can then be sent to an online server for emotion classification. Since the application only sends the statistics of the speech features to the server for processing instead of the entire speech utterance, the privacy of the user is better preserved and the bandwidth for transmission is reduced. A crowdsourcing test is also on-going using Amazon Mechanical Turk. In this test, users classify the speech utterances from the LDC and UGA datasets, so that we can compare the emotion classification results from the human coders with the results obtained from our system.

For the protein structure prediction problem, we are working to improve our proposed algorithm by adding void detection inside protein structures and optimizing the parameters used in our proposed algorithm, which may lead to a predicted structure closer to the native protein structure. We will further evaluate the performance of the proposed system on a larger protein dataset.

Bibliography

[1] "Emotional prosody speech and transcripts database from Linguistic Data Consortium (LDC)." http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2002S28.

[2] "Background noise samples used to construct the AURORA noisy speech database." http://www.ee.columbia.edu/~dpwe/sounds/noise/.

[3] P. C. Bagshaw, S. M. Hiller, and M. A. Jack, "Enhanced pitch tracking and the processing of F0 contours for computer aided intonation teaching," in *Proceedings of Eurospeech*, pp. 1003–1006, 1993.

[4] F. Plante, G. Meyer, and W. A. Ainsworth, "A pitch extraction reference database," in *Proceedings of Eurospeech*, pp. 837–840, 1995.

[5] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas, "EmotionSense: a mobile phones based adaptive platform for experimental social psychology research," in *Proceedings of the 12th int. conference on Ubiquitous computing*, pp. 281–290, 2010.

[6] D. Bitouk, V. Ragini, and N. Ani, "Class-level spectral features for emotion recognition," *Journal of Speech Communication*, vol. 52, no. 7-8, pp. 613–625, 2010.

[7] V. Sethu, E. Ambikairajah, and J. Epps, "Empirical mode decomposition based weighted frequency feature for speech-based emotion classification,"

in *Acoustics, Speech and Signal Processing, IEEE International Conference on*, pp. 5017–5020, 2008.

[8] T. Wang, W. Heinzelman, and A. Seyedi, "Minimization of transceiver energy consumption in wireless sensor networks with AWGN channels," in *46th Annual Allerton Conference on Communication, Control, and Computing*, pp. 62 –66, 2008.

[9] "UCSF Chimera molecular modeling system." `http://www.cgl.ucsf.edu/chimera/`.

[10] M. V. Shapovalov and R. L. Dunbrack, "A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions," *Structure*, vol. 19, pp. 844–858, 2011.

[11] M. A. Guvensan and A. G. Yavuz, "Hybrid movement strategy in self-orienting directional sensor networks," *Ad Hoc Networks*, vol. 11, pp. 1075–1090, 2013.

[12] M. E. Hoque, D. J. McDuff, and R. W. Picard, "Exploring temporal patterns towards classifying frustrated and delighted smiles," *IEEE Transactions on Affective Computing*, vol. 3, no. 3, pp. 323 –334, 2012.

[13] D. Glowinski, A. Camurri, G. Volpe, N. Dael, and K. Scherer, "Technique for automatic emotion recognition by body gesture analysis," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pp. 1–6, 2008.

[14] S. Jerritta, M. Murugappan, R. Nagarajan, and K. Wan, "Physiological signals based human emotion recognition: a review," in *Signal Processing and its Applications (CSPA), 2011 IEEE 7th International Colloquium on*, pp. 410–415, 2011.

[15] H. Ba, N. Yang, I. Demirkol, and W. Heinzelman, "BaNa: A hybrid approach for noise resilient pitch detection," in *IEEE Statistical Signal Processing Workshop (SSP)*, pp. 369 –372, 2012.

[16] N. Yang, H. Ba, W. Cai, I. Demirkol, and W. Heinzelman, "BaNa: A noise resilient fundamental frequency detection algorithm for speech and music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1833–1848, 2014.

[17] N. Yang, R. Muraleedharan, J. Kohl, I. Demirkol, W. Heinzelman, and M. Sturge-Apple, "Speech-based emotion classification using multiclass SVM with hybrid kernel and thresholding fusion," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pp. 455 –460, 2012.

[18] N. Yang, J. Yuan, Y. Zhou, I. Demirkol, W. Heinzelman, and M. Sturge-Apple, "How does noise impact speech-based emotion classification?," in *Designing Speech and Language Interactions Workshop, the ACM CHI Conference on Human Factors in Computing Systems*, 2014.

[19] N. Yang, J. Yuan, Y. Zhou, Z. Duan, W. Heinzelman, and M. Sturge-Apple, "Enhanced multiclass svm with thresholding fusion for speech-based emotion classification," *Submitted to Computer Speech and Language (Elsevier Journal)*.

[20] M. Sturge-Apple, P. T. Davies, D. Cicchetti, and L. G. Manning, "Interparental violence, maternal emotional unavailability and children's cortisol functioning in family contexts," *Developmental Psychology*, vol. 48, no. 1, pp. 237 –249, 2012.

[21] N. Yang, I. Demirkol, and W. Heinzelman, "Cross-layer energy optimization under image quality constraints for wireless image transmissions," in *Wire-*

less Communications and Mobile Computing Conference (IWCMC), 2012 8th International, pp. 1000–1005, 2012.

[22] N. Yang, I. Demirkol, and W. Heinzelman, "Motion sensor and camera placement design for in-home wireless video monitoring systems," in *IEEE Global Telecommunications Conference*, pp. 1–5, 2011.

[23] J. C. Latombe, "Protein structure similarity," in *Lecture notes for CS273 Structure and Motion in Biology at Stanford University*.

[24] D. Lee, O. Redfern, and C. Orengo, "Predicting protein function from sequence and structure," *Nature Reviews Molecular Cell Biology*, vol. 8, pp. 995 –1005, 2007.

[25] B. Cardozo and R. Ritsma, "On the perception of imperfect periodicity," *Audio and Electroacoustics, IEEE Trans. on*, vol. 16, no. 2, pp. 159 – 164, 1968.

[26] J. H. Jeon, W. Wang, and Y. Liu, "N-best rescoring based on pitch-accent patterns," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 732–741, 2011.

[27] C. Wang, *Prosodic Modeling for Improved Speech Recognition and Understanding*. PhD thesis, Massachusetts Institute of Technology, 2001.

[28] Z.-H. Ling, Z.-G. Wang, and L.-R. Dai, "Statistical modeling of syllable-level F0 features for hmm-based unit selection speech synthesis," in *Proceedings of International Symposium on Chinese Spoken Language Processing*, pp. 144–147, 2010.

[29] S. Sakai and J. Glass, "Fundamental frequency modeling for corpus-based speech synthesis based on a statistical learning technique," in *Proc. of IEEE ASRU*, pp. 712–717, 2003.

[30] J. Woodruff and D. L. Wang, "Binaural detection, localization, and segregation in reverberant environments based on joint pitch and azimuth cues," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 21, pp. 806–815, 2013.

[31] O. W. Kwon, K. Chan, J. Hao, and T. W. Lee, "Emotion recognition by speech signals," in *Eighth European Conference on Speech Communication and Technology*, 2003.

[32] F. Al Machot, A. H. Mosa, K. Dabbour, A. Fasih, C. Schwarzlmuller, M. Ali, and K. Kyamakya, "A novel real-time emotion detection system from audio streams based on bayesian quadratic discriminate classifier for adas," in *Nonlinear Dynamics and Synchronization 16th Intl Symposium on Theoretical Electrical Engineering*, 2011.

[33] K. Chang, D. Fisher, and J. Canny, "AMMON: A Speech Analysis Library for Analyzing Affect, Stress, and Mental Health on Mobile Phones," in *2nd Intl Workshop on Sensing Applications on Mobile Phones*, 2011.

[34] Y. Yang, C. Fairbairn, and J. F. Cohn, "Detecting depression severity from vocal prosody," *Affective Computing, IEEE Trans. on*, vol. 4, no. 2, pp. 142–150, 2013.

[35] J. P. Bello, G. Monti, and M. Sandler, "Techniques for automatic music transcription," in *Intl Symposium on Music Information Retrieval*, pp. 23–25, 2000.

[36] M. Antonelli, A. Rizzi, and G. Del Vescovo, "A query by humming system for music information retrieval," in *Intelligent Systems Design and Applications (ISDA), 10th Intl Conference on*, pp. 586 – 591, 2010.

[37] S. Kim, E. Unal, and S. Narayanan, "Music fingerprint extraction for classical music cover song identification," in *Multimedia and Expo, 2008 IEEE Intl. Conference on*, pp. 1261 –1264, 2008.

[38] P. Cariani, "Neural Representation of Musical Pitch - MIT OpenCourseWare." `http://ocw.mit.edu/courses/health-sciences-and-technology/hst-725-music-perception-and-cognition-spring-2009/lecture-notes/MITHST_725S09_lec04_pitch.pdf`, 2009.

[39] A. M. Noll, "Cepstrum pitch determination," *Journal of the Acoustical Society of America*, vol. 41, pp. 293–309, 1967.

[40] M. R. Schroeder, "Period histogram and product spectrum: New methods for fundamental frequency measurement," *Journal of the Acoustical Society of America*, vol. 43, pp. 829–834, 1968.

[41] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *Proceedings of the Institute of Phonetic Sciences 17*, pp. 97–110, 1993.

[42] H. Ba, N. Yang, I. Demirkol, and W. Heinzelman, "BaNa: A hybrid approach for noise resilient pitch detection," in *IEEE Workshop on Statistical Signal Processing*, pp. 369 –372, 2012.

[43] M. J. Ross, H. L. Shaffer, A. Cohen, R. Freudberg, and H. J. Manley, "Average magnitude difference function pitch extractor," *Audio, Speech, and Language Processing, IEEE Trans. on*, pp. 353 –362, 1974.

[44] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *Journal of the Acoustical Society of America*, vol. 111, pp. 1917–1930, 2002.

[45] J. Liu, T. F. Zheng, J. Deng, and W. Wu, "Real-time pitch tracking based on combined SMDSF," in *Proc. of Interspeech*, pp. 301–304, 2005.

[46] D. Talkin, "A robust algorithm for pitch tracking (RAPT)," *Speech Coding and Synthesis*, 1995.

[47] L. R. Rabiner and R. W. Schafer, *Theory and Application of Digital Speech Processing*. Pearson, 2011.

[48] T. W. Parsons, "Separation of speech from interfering speech by means of harmonic selection," *Journal of the Acoustical Society of America*, vol. 60, pp. 911–918, 1976.

[49] X. Chen and R. Liu, "Multiple pitch estimation based on modified harmonic product spectrum," in *Proceedings of Intl Conference on Information Technology and Software Engineering*, pp. 271–279, 2013.

[50] S. Gonzalez and M. Brookes, "A pitch estimation filter robust to high levels of noise (PEFAC)," in *Proc. European Signal Processing Conf., Barcelona, Spain*, 2011.

[51] Z. Jin and D. Wang, "HMM-based multipitch tracking for noisy and reverberant speech," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 19, no. 5, pp. 1091–1102, 2011.

[52] F. Huang and T. Lee, "Pitch estimation in noisy speech using accumulated peak spectrum and sparse estimation technique," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 21, no. 1, pp. 99 –109, 2013.

[53] M. Wu, D. Wang, and G. J. Brown, "A multipitch tracking algorithm for noisy speech," *Speech and Audio Processing, IEEE Trans. on*, vol. 11, no. 3, pp. 229–241, 2003.

[54] W. Chu and A. Alwan, "SAFE: A statistical approach to F0 estimation under clean and noisy conditions," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 20, no. 3, pp. 933 –944, 2012.

[55] A. von dem Knesebeck and U. Zölzer, "Comparison of pitch trackers for real-time guitar effects," in *Proc. of the 13th Int. Conference on Digital Audio Effects*, 2010.

[56] P. De La Cuadra, A. Master, and C. Sapp, "Efficient pitch detection techniques for interactive music," in *Proceedings of the Int. Computer Music Conference, La Habana*, 2001.

[57] Thomas O'Haver, Command-line findpeaks MATLAB function. `http://terpconnect.umd.edu/~toh/spectrum`.

[58] P. van Alphen and D. van Bergem, "Markov models and their application in speech recognition," in *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam*, pp. 1–26, 1989.

[59] D. Iskra, B. Grosskopf, K. Marasek, H. van den Huevel, F. Diehl, and A. Kiessling, "SPEECON - speech databases for consumer devices: Database specification and validation," in *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, pp. 329–333, 2002.

[60] B. Kotnik, H. Höge, and Z. Kacic, "Evaluation of pitch detection algorithms in adverse conditions," in *Proceedings of the Third International Conference on Speech Prosody*, pp. 149–152, 2006.

[61] I. Luengo, I. Saratxaga, E. Navas, I. Hernáez, J. Sanchez, and I. naki Sainz, "Evaluation of pitch detection algorithms under real conditions," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1057–1060, 2007.

[62] G. Seshadri and B. Yegnanarayana, "Performance of an event-based instantaneous fundamental frequency estimator for distant speech signals," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 19, pp. 1853–1864, 2011.

[63] "CMU Arctic Database." `http://www.festvox.org/cmu_arctic/`.

[64] "Generated noisy speech data and BaNa source code, WCNG website." `http://www.ece.rochester.edu/projects/wcng/project_bridge.html`.

[65] A. P. Varga, H. J. M. Steeneken, M. Tomlinson, and D. Jones, "NOISEX-92 study on the effect of additive noise on automatic speech recognition." `http://spib.ece.rice.edu/spib/data/signals/noise/`, 1992.

[66] L. R. Rabiner, M. J. Cheng, A. E. Osenberg, and C. A. McGonegal, "A comparative performance study of several pitch detection algorithms," *Acoustics, Speech, and Signal Processing, IEEE Trans. on*, vol. 24, pp. 399 – 418, October 1976.

[67] M. Wohlmayr, M. Stark, and F. Pernkopf, "A probabilistic interaction model for multipitch tracking with factorial hidden Markov models," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 19, no. 4, pp. 799–810, 2011.

[68] O. Babacan, T. Drugman, N. d'Alessandro, N. Henrich, and T. Dutoit, "A comparative study of pitch extraction algorithms on a large variety of singing sounds," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 7815–7819, 2013.

[69] "Download page for the SAFE toolkit." `http://www.ee.ucla.edu/~weichu/safe/`.

[70] X. Huang, A. Acero, and H.-W. Hon, *Spoken language processing*, vol. 15. Prentice Hall PTR New Jersey, 2001.

[71] D. Pearce, H.-G. Hirsch, and E. E. D. Gmbh, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ISCA ITRW ASR2000*, pp. 29–32, 2000.

[72] "Source code for the YIN algorithm." `http://audition.ens.fr/adc/`.

[73] "Source code for the Praat algorithm." `http://www.fon.hum.uva.nl/praat/`.

[74] "Source code for the PEFAC algorithm included in the VOICEBOX toolkit." `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`.

[75] "Source code for the Wu algorithm." `http://www.cse.ohio-state.edu/pnl/software.html`.

[76] M. G. Christensen and A. Jakobsson, *Multi-Pitch Estimation*. Morgan & Claypool Publishers, 2009.

[77] S. A. Raczynski, E. Vincent, and S. Sagayama, "Separation of speech from interfering speech by means of harmonic selection," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 21, pp. 1830–1840, 2013.

[78] M. Wu, D. L. Wang, and G. J. Brown, "A multipitch tracking algorithm for noisy speech," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 11, pp. 229–241, 2003.

[79] J. Wu, E. Vincent, S. A. Raczynski, T. Nishimoto, N. Ono, and S. Sagayama, "Polyphonic pitch estimation and instrument identification by joint modeling of sustained and attack sounds," *IEEE Journal of Selected Topics in Signal Process.*, vol. 5, pp. 1124–1132, 2011.

[80] "Freesound website for short pieces of music download." `http://www.freesound.org/`.

[81] Z. Duan, B. Pardo, and C. Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *Audio, Speech, and Language Processing, IEEE Trans. on*, vol. 18, no. 8, pp. 2121–2133, 2010.

[82] "BaNa Pitch Visualizer on Google Playstore." `https://play.google.com/store/apps/details?id=edu.rochester.ece.bana&hl=en`.

[83] K. R. Scherer, "What are emotions? and how can they be measured?," *Social Science Information*, vol. 44, no. 4, pp. 695–729, 2005.

[84] M. P. Black, A. Katsamanis, B. R. Baucom, C.-C. Lee, A. C. Lammert, A. Christensen, P. G. Georgiou, and S. S. Narayanan, "Toward automating a human behavioral coding system for married couples' interactions using speech acoustic features," *Speech communication*, vol. 55, no. 1, pp. 1–21, 2013.

[85] P. Kerig and D. Baucom, *Couple Observational Coding Systems*. Routledge, 2004.

[86] R. Bakeman, *Behavioral observation and coding, Handbook of research methods in social psychology*. Cambridge University Press, 1997.

[87] J. R. Bellegarda, "Data-driven analysis of emotion in text using latent affective folding and embedding," *Computational Intelligence*, vol. 29, no. 3, pp. 506–526, 2013.

[88] A. Goyal, E. Riloff, H. Daumé III, and N. Gilbert, "Toward plot units: Automatic affect state analysis," in *Proceedings of HLT/NAACL Workshop on Computational Approaches to Analysis and Generation of Emotion in Text (CAET)*, 2010.

[89] S. Özkul, E. Bozkurt, S. Asta, Y. Yemez, and E. Erzin, "Multimodal analysis of upper-body gestures, facial expressions and speech," in *Procceedings of the 4th International Workshop on Corpora for Research on Emotion Sentiment and Social Signals*, 2012.

[90] C.-H. Wu, C. Kung, J.-C. Lin, and W.-L. Wei, "Two-level hierarchical alignment for semi-coupled hmm-based audiovisual emotion recognition with temporal course," *IEEE Transactions on Multimedia*, vol. 15, no. 8, pp. 1880 – 1895, 2013.

[91] G. Huisman, M. Van Hout, E. van Dijk, T. van der Geest, and D. Heylen, "Lemtool - measuring emotions in visual interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013.

[92] J. Ang, R. Dhillon, A. Krupski, E. Shriberg, and A. Stolcke, "Prosody-based automatic detection of annoyance and frustration in human-computer dialog," in *Proceeings of International Conference on Spoken Language Processing*, pp. 2037–2040, 2002.

[93] P. Gupta and N. Rajput, "Two-stream emotion recognition for call center monitoring," in *INTERSPEECH*, pp. 2241–2244, 2007.

[94] C. M. Lee and S. S. Narayanan, "Toward detecting emotions in spoken dialogs," *Speech and Audio Processing, IEEE Transactions on*, vol. 13, no. 2, pp. 293–303, 2005.

[95] B. Schuller, G. Rigoll, and M. Lang, "Speech emotion recognition combining acoustic features and linguistic information in a hybrid support vector machine-belief network architecture," in *Acoustics, Speech, and Signal Processing (ICASSP), IEEE International Conference on*, vol. 1, pp. I–577, 2004.

[96] D. Tacconi, O. Mayora, P. Lukowicz, B. Arnrich, C. Setz, G. Troster, and C. Haring, "Activity and emotion recognition to support early diagnosis of psychiatric diseases," in *Pervasive Computing Technologies for Healthcare (PervasiveHealth), Second International Conference on*, pp. 100–102, 2008.

[97] R. Cowie, E. Douglas-Cowie, S. Savvidou, E. McMahon, M. Sawey, and M. Schröder, "'FEELTRACE': An instrument for recording perceived emotion in real time," in *ISCA Tutorial and Research Workshop (ITRW) on Speech and Emotion*, 2000.

[98] H. Bao, M.-X. Xu, and T. F. Zheng, "Emotion attribute projection for speaker recognition on emotional speech," in *Procceedings of Interspeech*, pp. 758–761, 2007.

[99] L. Qin, Z.-H. Ling, Y.-J. Wu, B.-F. Zhang, and R.-H. Wang, "Hmm-based emotional speech synthesis using average emotion model," in *Procceedings of Chinese Spoken Language Processing*, pp. 233–240, 2006.

[100] R. Barra-Chicote, J. Yamagishi, S. King, J. M. Montero, and J. Macias-Guarasa, "Analysis of statistical parametric and unit selection speech synthesis systems applied to emotional speech," *Speech Communication*, vol. 52, no. 5, pp. 394–404, 2010.

[101] S. Steidl, T. Polzehl, H. T. Bunnell, Y. Dou, P. K. Muthukumar, D. Perry, K. Prahallad, C. Vaughn, A. W. Black, and F. Metze, "Emotion identification for evaluation of synthesized emotional speech," in *Procceedings of Speech Prosody*, 2012.

[102] H. Kawanami, Y. Iwami, T. Toda, H. Saruwatari, and K. Shikano, "GMM-based voice conversion applied to emotional speech synthesis," in *Proceedings of Eurospeech*, 2003.

[103] V. N. Vapnik, *Statistical Learning Theory*. Wiley, 1998.

[104] M. Goudbeek, J. P. Goldman, and K. R. Scherer, "Emotion dimensions and formant position," in *INTERSPEECH*, pp. 1575–1578, 2009.

[105] S. Wu, T. H. Falk, and W.-Y. Chan, "Automatic recognition of speech emotion using long-term spectro-temporal features," in *Procceedings of the 16th International Conference on Digital Signal Processing*, 2009.

[106] T. Bänziger, S. Patel, and K. R. Scherer, "The role of perceived voice and speech characteristics in vocal emotion communication," *Journal of nonverbal behavior*, vol. 38, no. 1, pp. 31–52, 2014.

[107] D. A. Sauter, F. Eisner, A. J. Calder, and S. K. Scott, "Perceptual cues in nonverbal vocal expressions of emotion," vol. 63, no. 11, pp. 2251–2272, 2010.

[108] K. R. Scherer, "Vocal communication of emotion: A review of research paradigms," *Speech Communication*, vol. 40, no. 1-2, pp. 227–256, 2003.

[109] B. Schuller, G. Rigoll, and M. Lang, "Hidden markov model-based speech emotion recognition," in *Acoustics, Speech, and Signal Processing (ICASSP), IEEE International Conference on*, vol. 2, pp. II–1, 2003.

[110] S. Yun and C. D. Yoo, "Loss-scaled large-margin gaussian mixture models for speech emotion classification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 585–598, 2012.

[111] H. Tang, S. M. Chu, M. Hasegawa-Johnson, and T. S. Huang, "Emotion recognition from speech via boosted Gaussian mixture models," in *Multimedia and Expo (ICME), IEEE International Conference on*, pp. 294–297, 2009.

[112] S. Zhang, X. Zhao, and B. Lei, "Speech emotion recognition using an enhanced kernel isomap for human-robot interaction," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[113] R. Xia and Y. Liu, "Using i-vector space model for emotion recognition," in *Procceedings of Interspeech*, 2012.

[114] "Implementation of extracting MFCCs included in the VOICEBOX toolkit." `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`.

[115] N. H. de Jong and T. Wempe, "Automatic measurement of speech rate in spoken Dutch," *ACLC Working Papers*, vol. 2, no. 2, pp. 49–58, 2007.

[116] B. Schuller, B. Vlasenko, R. Minguez, G. Rigoll, and A. Wendemuth, "Comparing one and two-stage acoustic modeling in the recognition of emotion in speech," in *Automatic Speech Recognition Understanding, IEEE Workshop on*, pp. 596–600, 2007.

[117] L. Lee and R. C. Rose, "Speaker normalization using efficient frequency warping procedures," in *Acoustics, Speech, and Signal Processing (ICASSP), IEEE International Conference on*, vol. 1, pp. 353–356, 1996.

[118] U. Shrawankar and V. M. Thakare, "Adverse conditions and ASR techniques for robust speech user interface," *arXiv preprint arXiv:1303.5515*, 2013.

[119] B. Vlasenko, B. Schuller, A. Wendemuth, and G. Rigoll, "Combining frame and turn-level information for robust recognition of emotions within speech," in *INTERSPEECH*, pp. 2249–2252, 2007.

[120] M. Farrús, P. Ejarque, A. Temko, and J. Hernando, "Histogram equalization in SVM multimodal person verification," in *Proceedings of IAPR/IEEE International Conference on Biometrics*, 2007.

[121] J. Rong, G. Li, and Y.-P. P. Chen, "Acoustic feature selection for automatic emotion recognition from speech," *Information Processing and Management*, vol. 45, no. 3, pp. 315–328, 2009.

[122] I. Shafran, "A comparison of classifiers for detecting emotion from speech," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2005.

[123] C. M. Lee, S. S. Narayanan, and R. Pieraccini, "Combining acoustic and language information for emotion recognition," in *proceeding of 7th International Conference on Spoken Language Processing*, 2002.

[124] B. Roberto, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 537–550, 1994.

[125] "MATLAB implementation of mutual information." `http://www.mathworks.com/matlabcentral/fileexchange/14888-mutual-information-computation`.

[126] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," 2003.

[127] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., 1995.

[128] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods*, pp. 185–208, 1999.

[129] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2002.

[130] G. Wu and E. Y. Chang, "Class-boundary alignment for imbalanced dataset learning," in *Workshop on Learning from Imbalanced Datasets II, ICML*, pp. 49–56, 2003.

[131] M. Hoque, M. Yeasin, and M. Louwerse, "Robust recognition of emotion from speech," in *Intelligent Virtual Agents*, vol. 4133 of *Lecture Notes in Computer Science*, pp. 42–53, Springer Berlin Heidelberg, 2006.

[132] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Mchine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[133] F. Yu, E. Chang, Y. Xu, and H. Shum, "Emotion detection from speech to enrich multimedia content," in *Advances in Multimedia Information Processing - PCM*, pp. 550–557, 2001.

[134] J. Hernandez, M. E. Hoque, W. Drevo, and R. W. Picard, "Mood meter: Counting smiles in the wild," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 301–310, 2012.

[135] "Citizen Science project." `http://scienceforcitizens.net/`.

[136] D. Ververidis and C. Kotropoulos, "Emotional speech recognition: Resources, features, and methods," *Speech Communication*, vol. 48, no. 6, pp. 1162–1181, September 2006.

[137] M. F. Sabir, H. R. Sheikh, R. W. Heath, and A. C. Bovik, "A joint source-channel distortion model for JPEG compressed images," *IEEE Transactions on Image Processing*, vol. 15, pp. 1349–1364, 2006.

[138] J. H. G. Messier and R. Davies, "A sensor network cross-layer power control algorithm that incorporates multiple-access interference," *IEEE Transactions on Wireless Communications*, vol. 7, pp. 2877–2883, 2008.

[139] Y. Sankarasubramaniam, I. Akyildiz, and S. McLaughlin, "Energy efficiency based packet size optimization in wireless sensor networks," in *Proceedings of the 2003 IEEE International Workshop on Sensor Network Protocols and Applications*, pp. 1 – 8, 2003.

[140] M. Marijan, I. Demirkol, D. Maricic, and Z. I. G. Sharma, "Adaptive sensing and optimal power allocation for wireless video sensors with sigma-delta imager," *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2540 – 2550, 2010.

[141] W. C. Z. He and X. Chen, "Energy minimization of portable video communication devices based on power-rate-distortion optimization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 596 –608, 2008.

[142] M. Ritesh, S. Cui, L. Sanjay, and A. J. Goldsmith, "Modeling and optimization of transmission schemes in energy-constrained wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 1359–1372, 2007.

[143] S. Khan, S. Duhovnikov, and E. Steinbach, "Application-driven cross-layer optimization for video streaming over wireless networks," *IEEE Communications Magazine*, vol. 44, pp. 122–130, 2006.

[144] MICAz mote, http://www.openautomation.net, "Data sheet for the MICAz motes, Crossbow Technology Inc.."

[145] Microchip ZG2100M/ZG2101M WiFi transceiver module, http://ww1.microchip.com/downloads/en/DeviceDoc/70624A.pdf.

[146] J.-J. Gonzalez-Barbosa, T. Garcia-Ramirez, J. Salas, J.-B. Hurtado-Ramos, and J.-d.-J. Rico-Jimenez, "Optimal camera placement for total coverage," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 844 –848, May 2009.

[147] E. Hörster and R. Lienhart, "On the optimal placement of multiple visual sensors," in *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pp. 111–120, 2006.

[148] U. M. Erdem and S. Sclaroff, "Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements," *Comput. Vis. Image Underst.*, vol. 103, pp. 156–169, September 2006.

[149] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, "Senseye: a multi-tier camera sensor network," in *Proceedings of the 13th annual ACM international conference on Multimedia*, pp. 229–238, 2005.

[150] C. Yu and G. Sharma, "Camera scheduling and energy allocation for lifetime maximization in user-centric visual sensor networks," *Image Processing, IEEE Transactions on*, vol. 19, pp. 2042 –2055, August 2010.

[151] K.-J. Wong and D. K. Arvind, "SpeckMAC: low-power decentralised MAC protocols for low data rate transmissions in specknets," in *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pp. 71–78, 2006.

[152] Transceiver mote, http://www.cs.jhu.edu/ cliang4/public/datasheets/tmote-sky-datasheet.pdf.

[153] Motion sensor, http://www.allproducts.com/ee/irtec/Product-200881595949.html.

[154] Wireless camera, http://nservices.com/wireless.htm.

[155] "Protein data bank." http://www.rcsb.org/pdb/home/home.do.

[156] A. S. Fraenkel, "Protein folding, spin glass and computational complexity," in *In Proceedings of the 3rd DIMACS Workshop on DNA Based Computers*, pp. 175–191, 1997.

[157] J. Desmet, M. de Maeyer, B. Hazes, and I. Lasters, "The dead-end elimination theorem and its use in protein side-chain positioning," *Nature*, vol. 356, pp. 539–542, 1992.

[158] N. A. Pierce, J. A. Spriet, J. Desmet, and S. L. Mayo, "Conformational splitting: A more powerful criterion for dead-end elimination," *Journal of Computational Chemistry*, vol. 21, pp. 999–1009, 2000.

[159] P. Koehl and M. Delarue, "Application of a self-consistent mean field theory to predict protein side-chains conformation and estimate their conformational entropy," *Journal of Molecular Biology*, vol. 239, pp. 249–275, 1994.

[160] G. G. Krivov, M. V. Shapovalov, and R. L. Dunbrack, "Improved prediction of protein side-chain conformations with SCWRL4," *Proteins*, vol. 77, pp. 778–795, 2009.

[161] J.-J. Gonzalez-Barbosa, T. Garcia-Ramirez, J. Salas, J.-B. Hurtado-Ramos, and J.-d.-J. Rico-Jimenez, "Optimal camera placement for total coverage," in *IEEE International Conference on Robotics and Automation*, pp. 844–848, 2009.

[162] S. Chellappan, X. Bai, B. Ma, D. Xuan, and C. Xu, "Mobility limited flip-based sensor networks deployment," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 18, pp. 199–211, 2007.