Towards audio-assist cognitive computing : algorithms and applications

Liu, Ziyuan

2019

Liu, Z. (2019). Towards audio-assist cognitive computing : algorithms and applications.
Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/136992>

<https://doi.org/10.32657/10356/136992>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0
International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 22:02:18 SGT

NANYANG
TECHNOLOGICAL
UNIVERSITY

**TOWARDS AUDIO-ASSIST COGNITIVE
COMPUTING: ALGORITHMS AND
APPLICATIONS**

LIU ZIYUAN

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2019

**TOWARDS AUDIO-ASSIST COGNITIVE
COMPUTING: ALGORITHMS AND
APPLICATIONS**

LIU ZIYUAN

School of Computer Science and Engineering

A thesis submitted to Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Master of Engineering

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

2019/5/12

.....

Date

Liu Ziyuan

.....

Liu Ziyuan

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

2019/5/12

.....
Date

.....
Prof. Wen Yonggang

Authorship Attribution Statement

This thesis **does not** contain any materials from papers published in peer-reviewed journals or from papers accepted at conferences in which I am listed as an author.

2019/5/12

.....

Date

Liu Ziyuan

.....

Liu Ziyuan

Contents

Abstract	iv
Acknowledgement	v
Lists of Figures	ix
Lists of Tables	x
Chapter 1 Introduction	1
1.1 Background	1
1.1.1 Audio Content Recognition	2
1.1.2 Machine Learning Algorithms	3
1.2 Motivation and Objectives	5
1.2.1 Motivation	5
1.2.2 Objectives	7
1.3 Organization of the Thesis	8
Chapter 2 Literature Review	9
2.1 Audio Watermarking	9
2.1.1 Background and Applications	11
2.1.2 Algorithms	13
2.2 Audio Fingerprint	16
2.3 Machine Learning-Based Methods	20
Chapter 3 Core Technologies	23
3.1 Audio Tag	23
3.1.1 Contribution	24
3.1.2 Algorithm Design	24
3.1.3 Algorithm Implementation	26

3.1.3.1	Algorithm Version1	26
3.1.3.2	Algorithm Version2	27
3.1.4	Algorithm Performance Experiments	30
3.1.4.1	Evaluation Metrics	30
3.1.4.2	Silent Room Experiment	30
3.1.5	Segment Length Experiment	32
3.2	Audio Fingerprint	34
3.2.1	Contribution	34
3.2.2	Algorithm Design	34
3.2.3	Algorithm Implementation	37
3.2.4	Algorithm Performance Experiments	38
3.2.4.1	Storage Size Test	39
3.2.4.2	Performance Test	40
Chapter 4	Hey!Shake: Interactive TV Watching Android Application	44
4.1	Objectives	44
4.2	System Achitecture	44
4.2.1	Audio Fingerprint Workflow	46
4.3	Audio Fingerprint Workflow	48
4.4	Experiment Result	48
Chapter 5	Parking Loud: Smart Parking Lot Access Control system	51
5.1	Objectives	51
5.2	System Architecture	54
5.2.1	Workflow	55
5.2.2	Operation Mode	56
5.3	Experiments	56
Chapter 6	Conclusion and Future Works	59

6.1	Conclusion	59
6.2	Future Works	60

Abstract

Meaningful information hidden in the acoustic signals can be utilized by cognitive computing algorithms. The algorithms use them to improve the quality of services and applications. Inspired by this idea, we develop and optimize a series of applications based on cognitive computing algorithms. Two cognitive computing algorithms are developed: Audio Tag and Audio Fingerprint algorithms. The implementation and experiment results of the algorithms suggest that the information hidden in acoustic signals, either manually implanted or innate, can be utilized by proper techniques. The experiment results demonstrate that the audio tag and audio fingerprint algorithm have high accuracy and low time cost. The audio tag algorithm achieves 100% accuracy (recognition under 5 seconds), with loud noises existing in specific experiment environments. The audio fingerprint algorithm achieves over 95% accuracy (recognition under 5 seconds), with proper parameter settings. Based on the two core algorithms, two android applications are developed: Hey!Shake and Parking Loud application. They utilize these algorithms in the TV watching and parking lot access control scenarios and provide services with better quality, less hardware cost, and more convenience for users. The results of this research project confirm the possibility that we can improve the quality of multimedia services by digging into the often-overlooked acoustic information.

Keywords: Audio Watermark, Audio Tag, Audio Fingerprint, Acoustic Parking System, Acoustic Recognition and Feature Extraction.

Acknowledgement

I want to thank my mentor Prof. Wen Yonggang Dr. Ta Nguyen Binh Duong, and my colleagues from CAP research group who provided insight and expertise that greatly assisted the research, especially during the thesis writing and revising phase.

Also, I'd like to thank Xia Wenfeng, Yang Zhutian, and Huang Xingwei for assistance with the experiment design and experimental data collection, and Hu Weizheng for comments that significantly improved the manuscript.

At last, I'd like to thank my beloved girlfriend, Iris, who support me through all the hardness and dilemmas during my master's studying period. Without her, I am not able to make it through.

List of Figures

Figure 1.1	System architecture of an ACR System. (a) In reality, preprocessed audio content is played. The smartphones record the audios and upload them to our server for further processing and information retrieval (b) In our system workflow; it's a fusion of AT and AF algorithms. They share the recording and transmission phases. At the server, uploaded data is decomposed and analyzed separately	4
Figure 2.1	Audio Watermarking Process. With this embedding procedure, we can transform a certain watermark number into a kind of audio signals. The algorithm completes the transforming process by altering bits at fixed positions in each data bytes. In this way, when the receiver devices record the signal, the number can be properly decoded and used for the following workflow.	10
Figure 2.2	Three crucial factors of audio watermarking techniques. The trade-off between them marks the difference and the unique strength of each algorithm.	11
Figure 2.3	The algorithm process of spatial domain audio watermarking algorithms.	13
Figure 2.4	Power Spectral Density Diagram. Adapted from [1] by Al-salami, Mikdam AT and Al-Akaidi, MARWAN M	14
Figure 2.5	The algorithm process of frequency domain audio watermarking algorithms.	15

Figure 2.6	The typical pipeline of the audio fingerprint algorithm consists of three steps. First, a set of fingerprints are extracted from the query. Then the query is compared with the databases mentioned above. The database is usually partitioned to improve matching proficiency. Finally, a temporal alignment step is applied to the most similar matches in the database. .	19
Figure 2.7	SoundNet Network Architecture. It consists of two networks: 1) Teacher network. 2) Student network. Pretrained models are applied here as a teacher network. And the purpose of training student network is to make student network capable of generating similar features from raw waveform data. Adapted from [2] by Aytar, Yusuf and Vondrick, Carl and Torralba, Antonio	21
Figure 3.1	An example spectrogram of embedded Audio Tag Signals generated by Audio Tag Algorithm V1. As we can see, our signal stays above the usual audio content frequency range. Hence the content of video/audio won't be able to affect its performance. Also, this means that low-frequency noises, the most common kinds in our daily life, will not affect the recognition process.	26
Figure 3.2	Audio masking occurs when a sound possesses a relatively strong amplitude. In the frequency domain, it masks audio signals with similar frequency values and less amplitude. In the time domain, it masks audio signals which are present immediately preceding or following the strong signals. Adapted from [3], by online website "Hephaestus Audio"	28

Figure 3.3	An example spectrogram of embedded audio tag signals generated by audio tag algorithm version 2. As we can see, our signal stay above the usual audio content frequency range. Hence the content of videos/audios won't be able to affect its performance. Also this means that low-frequency noises, the most common kinds in our daily life, will not affect the recognition process.	29
Figure 3.4	Experiment result of min/mean/max recognition time cost in silent room environment. The recognition time cost increases as the distance between the sender and the receiver increases.	31
Figure 3.5	Experiment result of Segment Length Experiment. The recognition time cost increases as the distance between the sender and the receiver increases and it drops rapidly when the segment length decreases.	33
Figure 3.6	An example spectrogram with local amplitude maxima marked with dark blue dots. These local maxima are the foundation of generating audio fingerprint features. Due to the fact they possess relatively higher amplitude that surrounding points in the spectrogram, they are more robust and hard to be masked when facing noises attacks.	35
Figure 3.7	Zoomed Audio Fingerprint Feature	36
Figure 3.8	Key Parameters of Audio Fingerprint Algorithm	37
Figure 3.9	Recognition accuracy of audio fingerprint presented as a bar graph grouped by segment length. The accuracy significantly increases at the 3 seconds threshold, and speed of growth slows down afterward.	40

Figure 4.1	The screenshot of the main Activity of Hey!Shake application. Activity is the term of android development with the same meaning of page on the internet. It consists of UI components and user interaction logic.	45
Figure 4.2	Work Flow of Audio Tag. First, the algorithm combines the audio tag signal and video/audio contents by wiping off the near-ultrasound frequency audio components in the original data and inserting the audio tag signal.	46
Figure 4.3	Work Flow of Audio Fingerprint	48
Figure 4.4	Experiment Result of min/mean/max recognition time cost in Simulated living room environment. The recognition time cost increases as the distance between the sender and the receiver increases.	49
Figure 5.1	The screenshot of the main Activity of Parking Loud application.	52
Figure 5.2	Data flow of Parking Loud system. First, users register and provide their basic information. I take these user profile information to generate audio tag signals. Users can click to play these signals when they 're entering/exiting parking lots. The receiver devices at the entrance/exit catch and analyze these signals to know who is it interacting with and conduct corresponding operations correctly.	53
Figure 5.3	Experiment Result of min/mean/max recognition time cost in Real NTU Parking Lot environment. The recognition time cost increases as the distance between the sender and the receiver increases.	57

List of Tables

Table 3.1	Parameters of Audio Tag Algorithm Version 1	26
Table 3.2	Parameters of Audio Tag Algorithm Version 2	29
Table 3.3	Result of file size comparison experiment. The most common WAV format is uncompressed audio files. They are encoded in Linear Pulse Code Modulation(LPCM). Without the compression mechanism, the WAV format files take a lot of storage space. The MP3 and audio fingerprint format take about the same amount of storage space. This show that audio fingerprint can provide good compression of audio files while still keep crucial information of its content.	39
Table 3.4	Performance experiment result of audio fingerprint. Five groups of experiments are conducted in a simulated living room environment. Each group has a fixed minimal matching count. Then I alter the segment length from 2 seconds to 5 seconds and record all the result to calculate accuracy, error rate, miss rate and response time, where 1) accuracy is the percentage of correct recognized segments. 2) error rate it the percentage of wrongly recognized segments. 3) miss rate is the percentage of segments that can not be recognized. 4) response time is the average time cost of a single recognition	41

Chapter 1

Introduction

In this chapter, I'll present the background, motivation, and objectives of my research. The purpose of my work is exploring the possibility of utilizing hidden channel information and cognitive computing algorithms to enhance the quality of existing audio-based services and applications.

1.1 Background

Two popular series of cognitive computing algorithms are used to utilize hidden channel information: the traditional Audio Content Recognition (ACR) algorithms, and machine learning algorithms. The hidden channel information is a kind of information transmitted without the awareness of listeners. To avoid getting detected by the listeners, the changes made to the audio contents need to be either subtle or beyond the frequency range of the human hearing system. These changes are generated according to the information to be transmitted. By applying encoding algorithms, the receivers can correctly decode the received data. In the following subsections, I'll introduce how audio content recognition and machine learning algorithms work in this process.

1.1.1 Audio Content Recognition

Services and applications are developed based on audio content recognition algorithms. Digital multimedia equipment, like digital screens and sound-boxes, are installed everywhere. They continuously emit audio signals, which contain an enormous amount of information that can be utilized by ACR algorithms. Traditionally, there are no interactions between the users and audio contents. Users usually just watch or hear them for a while and leave it behind. With ACR algorithms, customers can learn more about the audio contents they like and get further information once the ACR algorithms finish the recognition process.

However, the quality of ACR applications and services is limited by noises and preprocessing operations of the playing devices/platforms. The performance of ACR applications heavily depends on the quality of the audio content. When the noises are too loud for the ACR algorithms to extract crucial information from the audio contents, no correct results can be retrieved. Another critical factor in the performance of ACR algorithms is the acoustic preprocessing process. Some online streaming service providers(such as YouTube, Netflix) apply low-pass filters for reasons like space-saving and speeding up data transmission. This inappropriate preprocessing disables ACR algorithms that utilize high-frequency components of the target audios.

Substantial efforts are made to overcome the obstacles mentioned above. Currently, there are 2 main series of ACR algorithm. Audio Tag(AT) represents one of them that utilize near-ultrasound frequency channel (18k-20kHz). Audio Tag technique achieves ACR functionality by embedding a high-frequency AT signal into original audio contents. Then the embedded signal is analyzed and decoded into a corresponding number under specific protocols. In this way, the Audio Tag technique can do real-time analysis of the audio contents. Once

the decoding phase finishes, the number is sent to the server to retrieve related information. Audio watermarking algorithms are robust against noise. Because in daily life, there are not many high-frequency noises since they are very disturbing and can be harmful to the human hearing system.

The other series of ACR algorithms is called Audio Fingerprint (AF), which analyzes low-frequency (0k-6kHz) audio components. AF algorithm extracts acoustic features(i.e., fingerprints) from the spectrogram. The acoustic features always come from audio components with lower-frequency so AF algorithms won't be affected by filtering or preprocessing of the original audio contents. As we can see, AT and AF algorithms all have their strength on certain occasions. However, their drawbacks are also apparent. Due to bit-rate and hardware issues, the high-frequency signal always gets filtered, which effectively disables AT algorithms. In contrast, audio fingerprint can fit on various occasions, but its performance shall significantly fall when applied in noisy environments.

1.1.2 Machine Learning Algorithms

The combination of machine learning algorithms and acoustic signal processing can be utilized to create applications that can't be developed by merely using traditional acoustic signal processing algorithms. The huge amount of data generated every day stimulates the improvement of machine learning, deep learning and related applications. Originally, the analysis of acoustic signals is heavily based on predefined features. There are many classic choices, such as FFT arrays [4] and Mel-Frequency Cepstral Coefficients(MFCC) [5]–[7]. Though these features work great in most acoustic research domains, deep learning managed to generate better features to represent characteristics of the audio signals, even without human knowledge. In recent years, many researchers have been trying various kinds of

(a) Real System

(b) Workflow

Figure 1.1: System architecture of an ACR System. (a) In reality, preprocessed audio content is played. The smartphones record the audios and upload them to our server for further processing and information retrieval (b) In our system workflow; it's a fusion of AT and AF algorithms. They share the recording and transmission phases. At the server, uploaded data is decomposed and analyzed separately

machine learning structures on the acoustic feature extraction problem. Among them, CNN is proved to have great efficiency and performance. There are many works like SoundNet and CNN Architecture for large scale audio classification proposed by Google team. They all take spectrogram data as input and generate high-level representation(i.e., features) of acoustic signals by running the input through the CNN layers. Afterward, these features are used for classification tasks. In most scenarios, the performances of the experiments turn out to be better than those using traditional audio features.

1.2 Motivation and Objectives

In this section, I'm going to present the motivation and objectives of my research.

1.2.1 Motivation

People have wondered about how to utilize technologies better to improve the user experience in our daily activities, like TV-watching and paying parking lot fees. These activities have a common feature that people do them almost the same way when they are invented.

Since the television was invented in late 19th century, audiences have been watching it in the same posture for over a hundred years. In the past few decades, developers and researchers in the television industry mainly focused on its display performance of the multimedia contents and regarded television as merely the carrier of the contents. With the rapid improvement in computer science, people can get in touch with tons of new content every day, which effectively makes the traditional TV-watching experience not attractive anymore. If audiences like the actor/actress or the clothes worn by them on TV, they can't conveniently acquire

further information that can be provided by the video-streaming applications on smartphones and tablets. Also, for parking lot scenarios, most drivers still have to drive towards the gate and pay their parking fees by cash or via e-payment methods. In developing countries/regions, the paying process is even completed by human employees. The two examples I just describe indicate the need to improve the user experience of human's daily events by applying proper technology is urgent.

Researchers and programmers have attempted to solve the problems in the past [8]–[11]. These works include a staffed toll booth, semi-auto control system, and auto-charge parking system. Among them, we notice the parking lot control system that is widely deployed in Singapore [9]. The system requires Radio Frequency Identification(RFID) [12] emitters and receivers to form the parking lot access control system. However, the cost of this system is also high. The RFID antenna is rather expensive itself, let alone every car owners need to install an in-vehicle Unit(IU) by paying \$150 to Singapore Land Transport Authority(LTA). These costs take a heavy toll on both the drivers' side and the managers' side. This drawback effectively prevents developing countries and regions from deploying this system.

The utilization of acoustic hidden channel information is the right solution to the problem. The reasons for choosing to use audio information are two folds. First, embedding acoustic hidden channel information data is somewhat accessible and insusceptible to users. Many characteristics of the human hearing system can be utilized to help with the embedding process, such as acoustic masking and audible frequency range. Only the receiver devices can recognize and extract the information while users cannot hear the difference. Besides, generating acoustic signals can currently be done by most devices with limited computing resources required, which effectively lowers the cost of the necessary hardware. Secondly, audio-based algorithms require little effort from users. Unlike visual-

based algorithms, audio-based algorithms don't require users to raise their phones to take photos or do other specific actions. Users only need to click to start the recording process and wait for the recognition result to be retrieved. Mobile devices have advantages in handling users' inputs and presenting information to users. These advantages can help traditional appliances to interact with customers. Microphones that are installed in both kinds of devices also support the application of acoustic cognitive algorithms in various scenarios.

1.2.2 Objectives

The main objective of my thesis is to **utilize the hidden channel information by applying cognitive computing algorithms to improve the quality of existing services and applications.**

- To solve the problems raised in the motivation section, I plan to design algorithms to encode the information to acoustic signals and decode them when needed. Additionally, two viable systems are built based on the algorithms targeting the TV-watching and parking scenarios to prove their abilities to improve the user experience of daily events.
- To prove the improvements and the value of the projects, I plan to compare my algorithms and systems with existing ones on the following measures:
 - Recognition Time
 - Recognition Accuracy
 - Storage Efficiency
- To give guidance in the relative research domain, further discussion will be made, covering the alternative designs as well as improvements of the

algorithms and the possibility of utilizing machine learning techniques to improve the performance.

I'll first present the details of the algorithms and then present two android mobile applications to provide improved services by utilizing audio-based algorithms. The experiment results are presented responsively.

1.3 Organization of the Thesis

The thesis is organized as follows:

1. Chapter 1 introduces the background, motivations, and objectives of my research project: Towards Audio-Assist Cognitive Computing: Algorithms and Applications
2. Chapter 2 presents the literature review result on acoustic cognitive computing
3. Chapter 3 demonstrates the details of the core technologies the applications are built on
4. Chapter 4 introduces the implementation detail and experiment results of Hey!Shake Application
5. Chapter 4 introduces the implementation detail and experiment results of Parking Loud Application
6. Chapter 6 concludes the thesis and introduces ideas for future work.

Chapter 2

Literature Review

Cognitive computing algorithms are used to improve the quality of existing services further by utilizing the hidden channel information. Two different levels of requirements are proposed to achieve the goals of my research:

- *Content recognition.*

The first requirements are called content recognition, which means the adopted algorithms must be able to recognize which pre-stored episode contains the uploaded audio snippet.

- *Playback Localization.*

Playback localization is the second requirement. To accurately know what users crave, we need to know what they are watching at that moment. Locating the real-time play position requires the capability to locate the playback location based on the uploaded audio snippet.

In this chapter, I will present the survey result on several prevalent acoustic analyzing algorithms that meet parts or all of the requirements.

2.1 Audio Watermarking

Audio watermarking algorithms are invented to protect the copyright of intellectual creations. The lack of identity, restrictions, and surveillance has encouraged

Figure 2.1: Audio Watermarking Process. With this embedding procedure, we can transform a certain watermark number into a kind of audio signals. The algorithm completes the transforming process by altering bits at fixed positions in each data bytes. In this way, when the receiver devices record the signal, the number can be properly decoded and used for the following workflow.

many kinds of malfeasance, among which the proliferation of digital content is one of the most frequent ones. Hence, audio watermarking is invented to give content providers the ability to protect their products. Simply adding pre-defined watermark signals is determined to deteriorate the quality of the contents. One of the most important requirements of watermarking is adding an imperceptible identifier that can tell us enough information for identification while the quality of the contents remains the same. And audio watermarking is a sub-concept of watermarking that mainly focuses on audio file protection.

The MPEG-1 level3(mp3) format is the most widely adopted audio file format. For reasons mentioned above, audio piracy has become a severe problem for the audio recording industry. Audio watermarking algorithms work by embedding some kind of extra data into the original data source. In the case of audio watermarking, the additional data to be embedded is usually a sequence of data. Base on this watermarking data, numerous applications can be developed. In the following subsections, I'll present an overview of the applications and various

Figure 2.2: Three crucial factors of audio watermarking techniques. The trade-off between them marks the difference and the unique strength of each algorithm.

algorithms related to the audio watermarking.

2.1.1 Background and Applications

Audio watermarking algorithms are designed for information transportation in the hidden channels to achieve anti-piracy or content identification. The main purposes are divided into two categories: proof of ownership and enforcement of usage policies [13]. Proof of ownership means that this kind of watermarking scheme can provide information to prove or indicate the owner of the protected digital contents. In this way, the consumer applications who are not evil can tell if the current user is not the same as the one indicated by the watermarking. Enforcement of usage means that this kind of watermarking scheme can provide information to consumer applications to prevent illegal duplication or violation in usage policy. In the next subsection, I'll introduce some audio watermarking algorithms. All of the audio watermarking schemes can be measured from several different perspectives:

- Robustness:

Robustness describes the reliability of watermark detection after the original data has been processed [14].

- Security:

Security reflects the capability of watermarking when against external attacks [15].

- Transparency:

Transparency describes how clear can human hear the audio watermarking signals.

- Complexity:

Complexity describes the difficulty of encoding/decoding of the audio watermarking signals

- Capacity:

Capacity describes the audio watermarking algorithms' ability to transfer data. This concept is rather similar to bit-rate.

There are some standards of objectives and evaluation of watermarking. The Recording Industry Association of America created the Secure Digital Music Initiative(SDMI) that intends to "develop open technology specifications for protected digital music distribution" [16]. Based on these standards, many applications are developed.

Besides anti-piracy, audio watermarking algorithms are also used for data transportation by utilizing the hidden channel communication technique.

Figure 2.3: The algorithm process of spatial domain audio watermarking algorithms.

2.1.2 Algorithms

There are three kinds of audio watermarking algorithms, the Spatial domain watermarking, the Frequency domain watermarking, the Hybrid domain watermarking

- *Spatial Domain*

The spatial domain audio watermarking algorithms directly embed watermarking signals into the audio content. This category of the watermarking algorithm is easy to implement and requires less computational power in comparison with the frequency domain methods and hybrid domain methods. Spatial domain watermarking algorithms include: LSB [17] replacement, echo hiding [18], phase coding [19], spread spectrum [20] and patch work [21].

Bassia proposes an audio watermarking algorithm, presented in [22], [23]. In this system, the audio watermark signal is modulated using the payload audio data, after which a low-pass filter is applied to the watermarking signal to reduce the noises and distortions that may influence the encoding and decoding phases. The audio watermark signal is added to the original audio signals repeatedly. The random seed of the chaotic sequence generator is the secret key chosen by the watermarking system to improve the security

Figure 2.4: Power Spectral Density Diagram. Adapted from [1] by Alsalami, Mikdam AT and Al-Akaidi, MARWAN M

level.

Figure 2.4 illustrated the Power Spectral Density(PSD) of the watermarked signals. The result shows that the audio watermark signal shaped by the algorithm has smaller PSD compared to the original audio signal so that it is inaudible to humans. According to the paper, this watermarking system is robust against attacks of time-shifting and cropping. The match is calculated for all possible circular shift to ensure synchronization between the probe signal and the original audio signal.

- *Frequency Domain*

The frequency-domain audio watermarking algorithm conducts frequential transformation before the embedding process. It's a more complex kind of audio watermarking algorithms compared with spatial domain ones. More

Figure 2.5: The algorithm process of frequency domain audio watermarking algorithms.

computation power is required in trade with more robustness against attacks. Frequency domain audio watermarking algorithms include: FFT [24], DCT [25], DWT [26] and SVD [27].

The algorithm process is illustrated in Fig. 2.5. The input signal is first transformed to the frequency domain where the watermark is embedded, the resulting signal then goes through inverse frequency transform to get the watermarked signal as output.

There are several methods for the frequency domain audio watermarking transformation. Cox and et al [28] proposed a methods based on spread spectrum technique. The watermark signals are designed to be a narrow-band audio signal. In this way, the signal energy of the watermark signals is spread over the frequency components in the whole transmission channel, so that the signal energy is small enough to be imperceptible on every single frequency. In Cox's theory, the cover signals(i.e., the media contents) are considered as the channel to transmit data. The watermark signal, attacks, and occasional distortions will all be transmitted through it. The authors claim that the watermark should be placed in perceptually significant regions of the cover signal to gain robustness against the noises and intentional attacks.

Arnold and et al [29] propose another method in 2000. This method utilizes

Fourier Transformation and patchwork algorithm [30]. In this method, the watermark signal is broken into frames, each of which embeds one data bit by transforming the data into the frequency domain using Discrete Fourier Transformation(DFT). A pattern is selected according to the embedded data bits, and the alteration made to the frequency domain coefficients must be inaudible to humans. All the parameters are driven from the psychoacoustics models [31] and then reshaped for each signal frame. The detection process is the inverse version of the embedding process. Further works have been done to improve the performance of the above audio watermarking system [21], [32] Kim, 2001]

- *Hybrid Domain*

The hybrid domain audio watermarking algorithms integrate the properties of both spatial and frequency domains so that the contradictory of the two kinds are balanced. Many characteristics are used to assess watermarking schemes. The most popular ones are robustness, imperceptibility, and capacity. According to the existing literature surveys, SVD is more robust in the wavelet domain without harming the audibility of the audio signals.

2.2 Audio Fingerprint

Audio fingerprint algorithms are designed to utilize the hidden channel information to achieve audio content recognition by comparing hash codes features generated on spectrogram data. A common example is a query-by-example recognition. This kind of application is useful when users hear something (such as music, drama), and want to know more about them. Shazam [33] and SoundHound [34] are good business cases in this industry. They are both popular music recognition applications on mobile devices. Other kinds of applications of

audio fingerprinting on mobile devices include copyright detection [35], [36], personalized entertainment, and interactive television watching without extra hardware [37].

Transplanting audio fingerprint application from PC to mobile devices introduces a set of new challenges that have never been seen before:

- *Latency*

Internet era began years ago. Back to the very start of its birth, users are used to the low bandwidth, connection quality, which in turn leads to poor user experience. However, with the rapid development in the IT industry, people now become very picky about the performance of the application they installed. Hence, to cater users' requests, the audio fingerprinting retrieval framework must be delicately designed to utilize all the computational power as well as all other kinds of resources to reduce the time users have to wait, i.e., the latency.

The total latency of audio fingerprinting algorithms consists of three parts: I. Local Processing Latency. II. Transmission Latency. III. Server Processing Latency. For the first part, the main cause of the latency is the processing of the recorded sample. The fewer operations we do on the phone, the shorter it is. Also, the sample length can affect this part too. The reasons are three folds: Firstly, if the sample itself is short, then the processing, such as data storing or compression, shall take less time. Secondly, the recording time is deemed as a part of waiting time though it's technically not. Lastly, the shorter sample possesses a smaller file size, which can reduce transmission latency. For the server processing latency, the analysis can be kind of complex because it's hard to analyze it alone. In most cases, the total computational tasks are independent upon environmental factors. So if we

do more tasks on the phone-end, there is less stress on the server-end. This is a trade-off, and it should be decided case by case by the developers.

- *Sample Length*

Current popular applications, such as Shazam and SoundHound, usually record audio samples with length over 10 seconds. In most cases, they can extract enough information to provide accurate recognition results. For other categories of audio fingerprinting applications, the sample length is different. For example, when it comes to copyright detection, the required sample length can grow to 30-60 seconds for retrieval [35].

- *Distortion*

In a real application environment, many factors can easily ruin the audio sample took by fingerprinting applications. And this kind of degradation or distortions tends to be more severe on mobile platforms. The mobile hardware is often the cause of them. Other causes can also be ambient noise in crowded places, time offsets, sampling errors and amplitude compression

The typical pipeline of the audio fingerprint algorithm consists of three steps. First, a set of fingerprints are extracted from the query. The fingerprints could be extracted at a uniform sampling rate, or only around points of interest in the spectrogram (e.g., spectrogram peaks proposed by Wang [38]). For mobile applications, this process has to be robust against ambient noises to make the query audio snippets more similar to the original audio contents stored in server-end databases.

Then the query is compared with the databases mentioned above. The reference tracks are thoroughly traversed to find substantial candidates. The database is partitioned to avoid pairwise comparison between the query and all of the reference tracks. The partitioning of the database is precomputed for the

Figure 2.6: The typical pipeline of the audio fingerprint algorithm consists of three steps. First, a set of fingerprints are extracted from the query. Then the query is compared with the databases mentioned above. The database is usually partitioned to improve matching proficiency. Finally, a temporal alignment step is applied to the most similar matches in the database.

database. Each partition is associated with a list of database songs (also called an inverted index). The partitioning on the database could be done by direct hashing of the fingerprints (e.g., a 32-bit fingerprint could be directly hashed into a table with 4 billion entries), Locality Sensitive Hashing or techniques based on Vector Quantization. The inverted file for each cell consists of a list of song IDs and the timing offsets at which the fingerprints appear. The timing information is used in the final step of the pipeline. Based on the number of fingerprints they have in common with the query probe from the inverted index, a shortlist of potentially similar database songs is selected from the database.

Finally, a temporal alignment step is applied to the most similar matches in the database. Techniques like Expectation-Maximization [39], RANSAC [40], or Dynamic Time Warping [41] are used for temporal alignment. In the case of linear correspondence (i.e., the tempo of the database and query songs are the same), Wang [38] proposes using a simple and fast technique that looks for a diagonal in the time-vs-time plot for matching database and query fingerprints. The existence of a strong diagonal indicates a valid match. The temporal alignment step is used to get rid of false positives and enables very high precision retrieval.

2.3 Machine Learning-Based Methods

Machine learning-based algorithms are designed to work with acoustic deep learning models to achieve audio content classifications. A large number of research domains have benefited from that, and audio based research is absolutely one of them. Currently, many of the acoustic machine learning methods are actually feature-driven. In the early stage of sound recognition research, traditional features are widely used. The popular choices are FFT [24], DCT [25], MFCC [42] and spectrograms. People believe that these kinds of features contain human knowledge are the best representation of acoustic signals. However, with the help of machine learning and deep learning, researchers start to generate new kinds of features by using the output of certain hidden layers. Then these features are used for various kinds of tasks, like classification or recognition. The test result turns out to be good with both general classifiers(SVM, GMM, etc.) and machine learning models(either supervised [43], [44] or unsupervised [45]). Recently, the state-of-the-art methods often apply Convolution Neural Network(CNN), a kind of machine learning network architecture that usually work with image data, to audio data. I have chosen two significant works for this survey.

Figure 2.7: SoundNet Network Architecture. It consists of two networks: 1) Teacher network. 2) Student network. Pretrained models are applied here as a teacher network. And the purpose of training student network is to make student network capable of generating similar features from raw waveform data. Adapted from [2] by Aytar, Yusuf and Vondrick, Carl and Torralba, Antonio

The first one is called SoundNet [2] proposed by an MIT research group. It's used for object and scene recognition. The brilliant idea of this work is that SoundNet creates two roles: teacher and student. The student is designed to learn knowledge from the teacher network through the training phase. For scene and object recognition, there are many mature pre-trained models, such as Place365 and ImageNet, that can be learned. Hence they are used as teacher models in this network. The student network needs to output the same kind of representation as a teacher network, which means the student network should be CNN too. In this case, the Fully Convolution Network is chosen to be the structure of the student network. It takes in raw waveform audio data and produces high-level representation with the same shape as the teacher network. Then the distance between the teacher network output and the student network output is measured by

Kullback-Leibler Divergence, which is invented to describe how one probability distribution is different from the second, reference probability distribution [46]. By narrowing the distance through the training phase, the student can learn to generate good acoustic features that are designed for specific tasks that, in this paper, are scene and object recognition. This work combines knowledge from transfer learning, cross-modal learning, and sound recognition and wisely shows us a possibility of multi-modal learning.

The other work is called "CNN Architecture for Large-Scale Audio Classification" [47], proposed by Google research group. CNNs have proven to have great performance in image classification in recent years. It's very natural to consider the possibility of its application in the audio research domain. However, traditional audio features, such as raw waveform, FFT arrays, and MFCCs is not ideal to be used as input of CNNs. So they choose to use the spectrogram of audio samples as the input in this work. They experiment many different CNN architectures: fully connected Deep Neural Networks(DNNs), AlexNet [48], VGG [49], Inception [50] and ResNet [51]. First, they do the experiments to test the impacts of parameters like the size of the label set, training size, and training steps. The result shows that many experiences found in the CV domain still works when it comes to the audio domain and larger training/label size can help to enhance the performance of the CNNs. Additionally, they conduct an experiment to compare the performance of the new features generated by CNNs with raw features(log-mel) when both of them are used for Audio Set [52] Audio Event Detection(AED). The result shows the performance is better when using embedding from the classifiers as input.

Chapter 3

Core Technologies

The core technologies include two algorithms: the Audio Tag algorithm and the Audio Fingerprint algorithm. In this chapter, I'm going to present the detail of the two algorithms, including overview, objectives, design terms, and results of experiments. The contributions of the algorithms mainly focus on the short recognition time and ability to fit in different environments. These two algorithms work together to provide robustness against both high-frequency and low-frequency noises. In this chapter, we propose an enhanced ACR algorithm named Unify ACR to combine the strengths of audio tag and audio fingerprint algorithms to avoid their weakness. Our motivation to develop this algorithm is that we found that the strength of these two kinds of ACR algorithms can overcome the limitations of each other. Hence we mainly focus on designing a new framework that can combine high-frequency signals embedding and fingerprint feature analysis, which enables Unify ACR to work against filtering and noises. Based on algorithms and system framework, two applications will be built to solve real-world problems.

3.1 Audio Tag

In this section, I'm going to present the details of the Audio Tag (AT) algorithm. This algorithm is designed for audio clip recognition. It is implemented by utilizing near ultra-frequency components of audio signals to avoid disturbing the

users. Two versions of the audio tag algorithm are developed. The first version is the prototype for demo. It simply adds a near-ultrasound signal generated by a tag number to the origin multimedia content. Then the receiver can decode the signal received and reconstruct the number a single bit at a time. However, the first version has rather poor performance and efficiency. So I develop the second version audio tag algorithm to improve its performance and capability of fitting into various kinds of application scenarios.

3.1.1 Contribution

The contribution of the audio tag algorithm is achieving data transmission through multimedia contents playing without letting the users be aware of it. To achieve that, we utilize the characteristic of the human hearing system that human beings with controlled energy levels cannot hear near-ultrasound acoustic signals. In the transmission process, the data will be encoded into an acoustic signal for content embedding. And the receiver devices can separate and decode the signal to acquire the origin data. The algorithm can work against noise attack and with low computing capacity requirements.

3.1.2 Algorithm Design

There are two main modules in the audio tag implementation: Audio Tag Adding module and Audio Tag Recognition module. The audio tag adding module takes a number as input and correspondingly outputs an audio tag signal. Certain standards have to be met to make sure receivers can easily and correctly recognize the audio tag signals. In the first version of the audio tag algorithm, I assign a set of parameters of the audio tag algorithm by an empirical definition. The

parameters are shown in Table 3.1. For a better understanding of the meanings of these parameters, I'll explain them with the help of figure 3.1.

- *Data Packet*

A data packet consists of several audio tag channels. Each channel represents a single bit of the audio tag number, which is represented in a certain radix.

- *Packet Length*

Packet length defines the length of a single data packet. For getting better analysis results, the equivalent count of audio samples should be close to a power of 2. The following formula can represent the relation between packet length T and the equivalent count of audio samples N_S :

$$N_S = \frac{T}{1000} \times 44100 \quad (3.1)$$

- *Channel Number*

Channel Number defines the number each data packet has. Due to the fact that total length of the data is fixed, the more channel number is, the shorter the complete audio tag signal will be.

- *Frequency Gap*

Frequency gap defines the vertical interval between frequencies that represent actual number.

- *Radix*

Radix N_R defines the numerical range a single channel represents.

- *Channel Width*

Channel width W_C can be defined by multiplying radix N_R and frequency gap N_G :

$$W_C = N_R \times N_G \quad (3.2)$$

Parameters	Packet Number	Packet Length	Channels Number	Frequency Gap	Radix
Values	3	200ms	3	20 Hz	41

Table 3.1: Parameters of Audio Tag Algorithm Version 1

Figure 3.1: An example spectrogram of embedded Audio Tag Signals generated by Audio Tag Algorithm V1. As we can see, our signal stays above the usual audio content frequency range. Hence the content of video/audio won't be able to affect its performance. Also, this means that low-frequency noises, the most common kinds in our daily life, will not affect the recognition process.

3.1.3 Algorithm Implementation

In this subsection, I'm going to present two versions of audio tag algorithms developed. The first version is developed as a prototype and can work properly. However, it has a rather slow recognition time. The second version is improved based on the drawbacks in the first version.

3.1.3.1 Algorithm Version1

In the prototype, we develop the audio tag algorithm version 1 based on the parameters listed in Table 3.1. The audio tag data consists of 3 parts: header, audio

tag number, and Cyclic Redundancy Check(CRC) [53]. The audio tag number is presented in a 6-bit unsigned number in base 41, which has a numeric range from 0 to 4750104240. Each data packet possesses three frequency channels. Every channel transmits one data bit by sending a 200 milliseconds long sinusoid with a frequency corresponding to the data value, as we mentioned in chapter 3. The CRC is used to check the validity of the received data. Considering the numeric range of the audio tag number, I adopt CRC16 to generate CRC codes and present them in a 3-bit number in base 41. So in total, the length of the audio tag signal is 3 data packets, which can just represent 9 data bits.

3.1.3.2 Algorithm Version2

Two main issues are found during the experiment of the version 1 algorithm: a) lack of header for localizing; b) errors introduced by recognizing 3 signals at the same time, such as audio masking [54]. So in version 2, I fix these issues and improve the performance of the algorithm. I'll explain them in detail in the following paragraphs.

In the first version, the lack of header for localizing make the recognizer module can't determine which packets contain data and which packets contain CRC values. The CRC check, in turn, forces the recognizer module to try the permutations of the received data packets. The calculation cost increase 3 times along due to that. Hence, to solve this problem, I add a header packet that is assigned with a frequency outside the data channel so that it won't be misrecognized as a regular data packet. The header is used to locate the start of the signal and save the redundant calculation in the recognition process. Once the recognizer module detects a header data packet, it can know the order to decode the received audio tag signal.

Figure 3.2: Audio masking occurs when a sound possesses a relatively strong amplitude. In the frequency domain, it masks audio signals with similar frequency values and less amplitude. In the time domain, it masks audio signals which are present immediately preceding or following the strong signals. Adapted from [3], by online website "Hephaestus Audio"

Figure 3.3: An example spectrogram of embedded audio tag signals generated by audio tag algorithm version 2. As we can see, our signal stay above the usual audio content frequency range. Hence the content of videos/audios won't be able to affect its performance. Also this means that low-frequency noises, the most common kinds in our daily life, will not affect the recognition process.

Parameters	Packet Number	Packet Length	Channel Frequency	Channel Number
Values	10	100ms	19000Hz	1

Table 3.2: Parameters of Audio Tag Algorithm Version 2

For the second issue, having to decode three data bits in one data packet brings problems. Even if the recognizer module correctly decodes 2 data bits right, it has to wait for 3 data packets and try to decode them completely right. This repeated process wastes much time. Also, audio masking occurs during this process and makes it even harder to get the correct result. To solve this problem, I unravel data packets and make each data packet only has one data bit. So in total, there are 10 data packets in a complete cycle of audio tag signal in audio tag algorithm version 2. The parameters and example spectrogram are shown below.

3.1.4 Algorithm Performance Experiments

3.1.4.1 Evaluation Metrics

Distance is the key parameters that will significantly affect the performance of our system. We first conduct the experiment in a silent room with different distance between the sender and the receiver. To measure how well our system work in near and mid range, we choose 1.5m, 3.5m and 6m as test distances. Then to prove our system can work in real environment, i.e. the parking lot, we choose 3 parking lots located in Nanyang Technological University and conduct the experiments at the entrance/exit to make sure the simulation is close to real application scenarios.

To measure how well the algorithm work, we record recognition correctness and time cost of the experiment. Based on all the records, minimal/mean/maximal recognition time and recognition accuracy are calculated. Since the signals are sent repeatedly, we can always get the correct result if given enough time. Hence, we define the correct recognition as the recognition correctly completed under 5 seconds. The minimal/mean/maximal recognition time are calculated according to conventions.

3.1.4.2 Silent Room Experiment

In silent room experiment, we use LeTV as the sender and use Samsung Note 8 as the receiver. We test at 3 different distances: 1.5m, 3.5m and 6m, each tested with 100 recognition. The receiver is placed in the middle line that is perpendicular to the TV surface and the internal microphones of the receiver is heading towards the TV. The test results are shown in Figure 3.4.

From the result, we can observe that the audio tag algorithm works stably

Figure 3.4: Experiment result of min/mean/max recognition time cost in silent room environment. The recognition time cost increases as the distance between the sender and the receiver increases.

in a quiet environment. Distance is a critical factor in the performance of the audio tag algorithm. As the distance increase, the recognition time increases as well. The phenomenon is easy to understand because that high-frequency signals' energy rapidly decays as they travel through the air. However, we can see that the maximal increment in recognition time is 0.2 seconds, between the 1.5m group and the 6m group. The increment is rather small, considering the distance is 3 times larger.

Also, we can state that the mean recognition time in each group is closer to the min time. This result indicates that, in most cases, the audio tag algorithm completes the correct recognition in a short period. We choose some of the audio samples where our system performs poorly and listen to it. Some impulsive noises can be heard in some of the samples. This kind of noise can produce the impulse onto the spectrogram as well, which can affect the near-ultra channel we need. Even though our system can still successfully recognize the tag signal in such a situation, we still suggest that avoiding this kind of environment for system deployment.

3.1.5 Segment Length Experiment

In addition to the silent room experiment, we try different segment lengths of the probe signals and test the performance, trying to find the balance between the short recognition time and the level of disturbances upon the young demographic who tends to have better hearing sensitivity. In that case, the audio watermarking signals can be squeaky and harmful.

In this experiment, we use the same experiment setting with the silent room experiment to eliminate the influence of the parameters. The receiver device records audio snippets with different watermark length of 50ms, 100ms, and

Figure 3.5: Experiment result of Segment Length Experiment. The recognition time cost increases as the distance between the sender and the receiver increases and it drops rapidly when the segment length decreases.

200ms. The experiment is conducted at a sender-receiver distance of 1.5m, 3.0m, and 6.0m. Accuracy is calculated for each experiment set by averaging 100-time tries.

From the result, we can observe that the audio tag algorithm works stably in 100ms and 200ms experiment sets. Segment Length is a critical factor in the performance of the audio tag algorithm. Longer tag segment length means higher frequency resolution. As the segment length decreases, the recognition time increases as well. This is easy to understand because, with worse frequency resolution, it's harder to map the frequency peaks to index value accurately. We can see that the sudden drop in the 50ms experiment set. This means the audio tag algorithms should work with the segment length parameter bigger than 50ms to maintain a rather good performance.

3.2 Audio Fingerprint

In this subsection, we are going to present the details of the Audio Fingerprint(AF) algorithm, including the description of the algorithm, implementation details of the algorithm in different programming languages, and the performance experiments result.

3.2.1 Contribution

The contribution of the audio fingerprint algorithm is to recognize the playing content based on matching with the pre-built multimedia database. This algorithm is designed to utilize low-frequency acoustic features. The frequency range is around 0 - 6k Hz. A feature extraction algorithm is implemented and all the contents(both uploaded sample). Also, we improve the hash and partition algorithm applied at servers to speed up the whole process. Compared to the original version, our fingerprint algorithm takes only one-fourth of the time to finish the same recognition job with the same accuracy.

3.2.2 Algorithm Design

The first version of the audio fingerprint algorithm is designed by referencing the python library [55] open-sourced by Dan Ellis, who works for Google and Columbia University. The "fingerprints" are locality-sensitive hashes generated from the spectrogram. The hashing process is done by calculating the FFT of the signal over the sliding windows of the song and spectrogram peaks. A very robust peak finding algorithm is needed. Otherwise, you'll have a terrible signal to noise ratio.

Figure 3.6: An example spectrogram with local amplitude maxima marked with dark blue dots. These local maxima are the foundation of generating audio fingerprint features. Due to the fact they possess relatively higher amplitude than surrounding points in the spectrogram, they are more robust and hard to be masked when facing noises attacks.

Figure 3.7: Zoomed Audio Fingerprint Feature

Finding these local maxima is a combination of a high pass filter (a threshold in amplitude space) and some image processing techniques to find maxima within the neighborhood, local maximum with only its directly adjacent pixels. The maxima are divided into strong peaks and poor peak. The poor peaks are ones that can't survive the noise of coming through speakers and a microphone.

If we zoom in even closer, we can begin to imagine how to bin and discretize these peaks. Finding the peaks itself is the most computationally intensive part, but it's not the end. Peaks are combined using their discrete-time and frequency bins to create a unique hash for that particular moment in the audio contents and creating a fingerprint. I developed an android application and a server-end audio fingerprint module to construct the whole system. The application is used to record and upload audio samples for a query with necessary preprocessing.

```

Options:
-d <dbase>, --dbase <dbase>      Fingerprint database file
-n <dens>, --density <dens>      Target hashes per second [default: 20.0]
-h <bits>, --hashbits <bits>      How many bits in each hash [default: 20]
-b <val>, --bucketsize <val>      Number of entries per bucket [default: 100]
-t <val>, --maxtime <val>          Largest time value stored [default: 16384]
-u <val>, --maxtimebits <val>      maxtime as a number of bits (16384 == 14 bits)
-r <val>, --samplerate <val>       Resample input files to this [default: 11025]
-p <dir>, --precomdir <dir>        Save precomputed files under this dir [default: .]
-i <val>, --shifts <val>           Use this many subframe shifts building fp [default: 0]
-w <val>, --match-win <val>        Maximum tolerable frame skew to count as a match [default: 2]
-N <val>, --min-count <val>        Minimum number of matching landmarks to count as a match [default: 5]
-x <val>, --max-matches <val>      Maximum number of matches to report for each query [default: 1]
-X, --exact-count                 Flag to use more precise (but slower) match counting
-R, --find-time-range             Report the time support of each match
-Q, --time-quantile <val>         Quantile at extremes of time support [default: 0.05]
-S <val>, --freq-sd <val>          Frequency peak spreading SD in bins [default: 30.0]
-F <val>, --fanout <val>           Max number of hash pairs per peak [default: 3]
-P <val>, --pks-per-frame <val>    Maximum number of peaks per frame [default: 5]
-D <val>, --search-depth <val>     How far down to search raw matching track list [default: 100]
-H <val>, --ncores <val>           Number of processes to use [default: 1]
-o <name>, --ofile <name>         Write output (matches) to this file, not stdout [default: ]
-K, --precompute-peaks           Precompute just landmarks (else full hashes)
-k, --skip-existing              On precompute, skip items if output file already exists
-C, --continue-on-error          Keep processing despite errors reading input
-l, --list                       Input files are lists, not audio
-I, --sortbytime                 Sort multiple hits per file by time (instead of score)
-v <val>, --verbose <val>         Verbosity level [default: 1]
-I, --illustrate                 Make a plot showing the match
-J, --illustrate-hpf             Plot the match, using onset enhancement
-W <dir>, --wavdir <dir>          Find sound files under this dir [default: ]
-V <ext>, --wavext <ext>          Extension to add to wav file names [default: ]
--version                        Report version number
--help                           Print this message

```

Figure 3.8: Key Parameters of Audio Fingerprint Algorithm

The audio fingerprint module processes the uploaded audio samples. The module resamples the uploaded audio samples and extracts the spectrogram from it to find the local maxima or "peak". Then these adjacent peaks will be combined to form a constellation mentioned in the paper [38] where the hash is generated. Features are compared to the feature datasets where all features are generated in the same process.

3.2.3 Algorithm Implementation

The implementation of the audio fingerprint algorithm is developed to work in both server and mobile devices in Python and Java programming language, respectively. There are many parameters to modify during the experiments. All the modifiable parameters are listed in Figure 3.8. I'm going to introduce some key parameters in this subsection.

- *Segment Length*

Segment time is the length of the recorded audio sample. The longer the

audio sample is, the more information it contains, which may, in turn, help with improving the recognition accuracy. In the experiments, I test the performance of the audio fingerprint algorithm with a segment time of 2s, 3s, 4s, and 5s. Longer segment time may make users reluctant to wait and stop using the service.

- *Fan-Out Count*

As shown in Figure 3.8, the hash features are generated by combining near-by peaks. The fan-out count is the number of neighbors a single peak reaches out. With larger fan-out count, the total number of hash features increases correspondingly. The more features extracted, the more information preserved in the dataset, and it can increase the recognition accuracy. But with more features, the time cost of the recognition increases as well. To deploy the service, the balance between time-cost and accuracy needs to be considered carefully.

- *Minimal Matching Count*

Minimal matching count decides the minimal matching features a correct matching must-have. Due to the traits of the features generation process, there are chances that there might be shared features between different audio files. By setting a larger minimal matching count, the misrecognition rate is effectively reduced.

3.2.4 Algorithm Performance Experiments

A series of experiments are conducted to measure the performance of the audio fingerprint algorithm, and the experiment results indicate that the audio fingerprint algorithm has high recognition accuracy with good storage efficiency.

Storage Type	Storage Size(MB)
WAV	1885
MP3	337
Audio Fingerprint	377

Table 3.3: Result of file size comparison experiment. The most common WAV format is uncompressed audio files. They are encoded in Linear Pulse Code Modulation(LPCM). Without the compression mechanism, the WAV format files take a lot of storage space. The MP3 and audio fingerprint format take about the same amount of storage space. This show that audio fingerprint can provide good compression of audio files while still keep crucial information of its content.

3.2.4.1 Storage Size Test

The result of the storage experiment shows that only saving the audio fingerprint feature takes significantly less storage then saving the raw WAV audio files. The dataset consists of 45 songs in WAV format. The comparison is conducted between 3 kinds of file storage formats: raw WAV, MP3, and audio fingerprint feature form.

As we can see from table 3.3, the audio fingerprint feature files take a much smaller amount of storage space than the WAV format, similar to the MP3 format. The WAV files possess the largest storage size of 1885 MB, approximately 5 times more than the MP3 and audio fingerprint file size. Mp3 and audio fingerprint's file size are in the same numeric scale. However, the MP3 media files can hardly be further compressed while the audio fingerprint features are binary data files, which can be even more storage efficient if existing compression algorithms are properly applied. Besides, the audio fingerprint features can be directly used for ACR problem and other audio-based problems while preprocessing is required

Figure 3.9: Recognition accuracy of audio fingerprint presented as a bar graph grouped by segment length. The accuracy significantly increases at the 3 seconds threshold, and speed of growth slows down afterward.

if the input data is stored in WAV/MP3 format. Developers can utilize these advantages even not in ACR problems, but also for data compression and feature extraction. In conclusion, we deem audio fingerprint features outperform other media file formats in terms of storage space efficiency.

3.2.4.2 Performance Test

According to Table 3.4, the baseline performance of the audio fingerprint algorithm achieves 90% accuracy with proper parameter setting. The experiment is conducted in a room with people casually talking to each other to simulate the living room application scenarios.

For the dataset, I download a subset of the YouTube8M dataset with audio

Minimal Count	Seg. Length	Acc.	Err. Rate	Miss Rate	Response Time
100	5 s	90.0%	8.7%	3.67%	305 ms
100	4 s	75.5%	16.5%	8.0%	271 ms
100	3 s	73.2%	18.6%	8.2%	248 ms
100	2 s	17.3%	27.1%	55.6%	153 ms
150	5 s	87.7%	1.6%	8.3%	329 ms
150	4 s	72.5%	4.0%	23.5%	257 ms
150	3 s	70.4%	3.0%	26.5%	246 ms
150	2 s	5.3%	2.3%	92.4%	142 ms
200	5 s	85.3%	0.3%	14.3%	264 ms
200	4 s	60.5%	0.5%	38.9%	205 ms
200	3 s	56.6%	0.6%	42.8%	195 ms
200	2 s	2.0%	0.1%	97.9%	128 ms
250	5 s	78.0%	0.3%	21.6%	280 ms
250	4 s	44.8%	0.2%	50.9%	193 ms
250	3 s	44.2%	0.2%	55.6%	182 ms
250	2 s	0.67%	0.0%	99.7%	121 ms
300	5 s	67.7%	0.3%	32.0%	277 ms
300	4 s	33.3%	0.3%	66.4%	215 ms
300	3 s	30.8%	0.2%	69.0%	201 ms
300	2 s	0.53%	0.0%	99.5%	109 ms

Table 3.4: Performance experiment result of audio fingerprint. Five groups of experiments are conducted in a simulated living room environment. Each group has a fixed minimal matching count. Then I alter the segment length from 2 seconds to 5 seconds and record all the result to calculate accuracy, error rate, miss rate and response time, where 1) accuracy is the percentage of correct recognized segments. 2) error rate it the percentage of wrongly recognized segments. 3) miss rate is the percentage of segments that can not be recognized. 4) response time is the average time cost of a single recognition

content for more than 28 hours. Distance between the test smartphone(as the recorder) and the LeTV(as the sound source) is fixed at 2.7 meters. For the minimal matching count, I choose 100, 150, 200, 250, 300 as experimental settings. For the segment length, I choose 2s, 3s, 4s, 5s as experimental settings. I record all the audio content in the dataset using the smartphone and upload them to test the accuracy and other metrics.

For performance evaluation, four metrics are applied to measure the performance:

- accuracy is the percentage of correct recognized segments.
- error rate it the percentage of wrongly recognized segments.
- miss rate is the percentage of segments that can not be recognized.
- response time is the average time cost of a single recognition.

Each experiment group has a fixed minimal matching count and 4 different segment length settings, ranging from two seconds to five seconds. From the experiment results, we can observe that better recognition accuracy can be achieved with longer segment length and an appropriately smaller minimal matching count.

The best recognition accuracy of 90% is achieved in the 100 minimal matchings count/5s segment length group. In the meantime, huge accuracy drops, averagely 20%, are observed between the 5s and 4s experiment group and the 3s and 2s experiment group, which set two thresholds for the segment length parameter. The response time drops as the minimal matching count decreases, and segment length increases. The high minimal matching count can trigger the pruning mechanism that we implement to speed up the feature matching procedure,

and longer segment length enlarges the feature input count, which enlarges the computational cost. The influence of segment length is significant, according to the table. However, this part of the time cost can be reduced by simply add more computational captivity. Lastly, when compared between the different group, we can observe that higher minimal count can lead to lower accuracy, error rate as well as response time, but the miss rate increase significantly. These characteristics can be utilized in the deployment of the audio fingerprint system. In realistic application scenarios, the priority order between precision, recall, and response time may differ from time to time. The user can implement their own ideal audio fingerprint system by tuning the parameters listed above, which proves the great flexibility possessed by our system.

Chapter 4

Hey!Shake: Interactive TV Watching Android Application

4.1 Objectives

The first application I designed is called Hey!Shake. Hey!Shake application is an android application developed to provide enhanced interactive TV watching experience. I intend to combine audio watermarking and audio fingerprint algorithms to balance each other's advantages and disadvantages. The audio tag system mainly utilizes the near ultra frequency channel, and the audio fingerprint system mainly utilizes the low-frequency channel. Either one of them can be affected by a single attack. However, when they are combined, it's much harder to disable the whole system. The main idea is simple. The mobile application takes charge of recording, simple acoustic processing, and uploading. Then the server analyzes the recorded audio sample and returns the result to the mobile application.

4.2 System Achitecture

The Hey!Shake application utilizes both audio tag and audio fingerprint algorithms to form a unified ACR system.

Figure 4.1: The screenshot of the main Activity of Hey!Shake application. Activity is the term of android development with the same meaning of page on the internet. It consists of UI components and user interaction logic.

Figure 4.2: Work Flow of Audio Tag. First, the algorithm combines the audio tag signal and video/audio contents by wiping off the near-ultrasound frequency audio components in the original data and inserting the audio tag signal.

4.2.1 Audio Fingerprint Workflow

The audio watermarking part is very straight forward. The main idea is to assign each multimedia content with a unique number as a tag. Before playing, this tag number is transformed into an audio signal and be embedded in the original data so that the content can and only be recognized by us. The system, namely Audio Tag(AT) System, consists of following parts:

- *Audio Contents Preprocessing.* In this part, we embed certain AT signals into target audio contents and transcode them with specific configurations to facilitate further operations. AT algorithms works in the following steps: First, we choose the tag number range and base N_t so that we can determine how many sub-signals shall be needed to represent a certain tag number. Then we choose a proper sub-signal length of T_0 . Then the AT signal $S(t)$ can be represented as:

$$S(t) = k \cdot f(t) + f_c \quad (4.1)$$

where f_c is the chosen variable to mark the start of the AT frequency Channel. Function $f(t)$ will calculate which digits of tag should be shown at moment t . The constant variable k depends on the width of the audio fingerprint frequency channel and the predefined base of the tag number.

$$k = \frac{\Delta f_c}{N_t} \quad (4.2)$$

To not disturb users' hearing systems, the frequency channel always is at the near-ultrasound part of the spectrogram. Hence, to make it work normally, the sample rate shall be set to 44100Hz, which is a traditional value. After that, we can put it together with the origin audio contents so that it can be analyzed by AT and AF algorithms afterward.

- *Recording.* In this part, the smartphone record the surrounding audio contents. We use little-endian PCM16 format to record and store our audio data with a 44100Hz sample rate and the 192k bitrate. If certain smartphones do not support this format, we respond manually to transcode and cater to our request. The processed data is uploaded to our server for information retrieval. Also, it's possible to do the feature extraction or AT analysis during this phase, but we temporally not consider that in this paper.
- *Decomposing.* The uploaded audio data goes through STFT processing. Then the spectrogram data can be used to find the strongest frequency component in each time step. Since high-frequency are rather rare to appear in our daily life, so it's safe to think that the strongest signal is our audio watermarking signal. At last, we can combine each time step's analysis result to recover the origin audio watermarking number.

Figure 4.3: Work Flow of Audio Fingerprint

4.3 Audio Fingerprint Workflow

The Audio Fingerprint part is a little bit complicated. I design this part based on Shazam fingerprint algorithm [38]. By applying this algorithm, the interactive TV application can acquire the ability to know the current playback location. The workflow of the audio fingerprint can be found in Figure 4.3. It's a simple feature matching. The feature used by this algorithm is generated in three steps. First, generate a spectrogram through STFT. Then find all the local energy maximum points, which are called peaks in the paper. Pair adjacent peaks to get the final feature. When in the matching phase, the server extracts the feature from the uploaded audio sample following the same process to compare with the database to get the matching result.

4.4 Experiment Result

The experiment results presented in Figure 4.4 indicate that Hey!Shake applications can work well in a simulated living room environment to help users catch their purchasing impulse while watching TV programs. The simulated experiment consists of LeTV as the sender, Samsung Note 8 mobile phone as the receiver, and recorded noises played in the background. The experiments are conducted at 3 different distances: 1.5m, 3.5m, and 6m to cover different distances between the

Figure 4.4: Experiment Result of min/mean/max recognition time cost in Simulated living room environment. The recognition time cost increases as the distance between the sender and the receiver increases.

users and the TV due to the difference in living room layout design. The receiver device is placed in the middle line that is perpendicular to the TV surface, and the internal microphones of the receiver are heading towards the TV. 100 recognitions are performed at each distance settings, and the min/mean/max time is recorded as the metrics.

From the result, we can observe that the Hey!Shake application can work stably in a simulated living room environment and maintain similar performance to baseline experiments against daily noise attacks. Compared to the silent room performance experiment result in table 3.3, the time cost slightly increases, averagely 0.1 seconds, which indicates that the algorithm can work well against noises and the interference of multimedia contents in TV watching scenarios.

In all the experiment settings, the 1.5m experiment has the best performance

in all metrics: 1.077s minimal recognition time, 2.265s mean recognition time, and 4.117s max recognition time. The application takes a long time to recognize the contents in experiments with further distance, but in total, the performance is stable. With a 6m range, the Hey!Shake application can cover all different kinds of living room and TV-watching situations. The average distance between the watching spot and the TV is 2.7m, which assures that users can use the Hey!Shake application from every position in the living room with good recognition time and service quality.

Chapter 5

Parking Loud: Smart Parking Lot Access Control system

5.1 Objectives

Parking Loud application is an android application developed to help developing countries/regions to upgrade their manually managed parking lots to automatic ones with low monetary and energy costs.

According to research conducted by Donald C. Shoup [56], [57], drivers spend a significant amount of time for parking activities every day. A standard parking procedure may include the following steps: cruising for vacant spaces, queuing for access, and actual parking. Among these steps, the cruising time cost heavily depends on traffic status; actual parking time cost depends on drivers' technique. Hence queuing time cost is the only choice to reduce total parking time effectively. In the traditional parking lot, tolling entrances/exits are usually assigned to a certain number of employees who are responsible for charging and surveillance. Using human power to do this kind of job has some obvious drawbacks. First of all, the processing speed is relatively slow. Also, for a human to complete a single charging, he/she have to finish all the following steps: calculating the correct parking fee, take the money and give the change back. During the procedure, the speed and accuracy gradually drop as the fatigue accumulates, which in turn causes severe queuing problems and dramatically increase tolling time. So it's

Figure 5.1: The screenshot of the main Activity of Parking Loud application.

Figure 5.2: Data flow of Parking Loud system. First, users register and provide their basic information. I take these user profile information to generate audio tag signals. Users can click to play these signals when they 're entering/exiting parking lots. The receiver devices at the entrance/exit catch and analyze these signals to know who is it interacting with and conduct corresponding operations correctly.

natural to consider a technological solution to upgrade those traditional parking lots.

Several kinds of innovating parking systems have been developed to ameliorate this conundrum. For example, Radio Frequency Identification(RFID) [12] technology is applied to control the car park access system in most parts of Singapore. The critical components of this system are the RFID antennas that are installed at the polling booth to work with the In-vehicle Units(IU). Drivers have to install the IUs behind the windshield of the vehicles to acquire parking lot privileges. To pay the parking fee, drivers need to drive to the exit and wait for the completion of the paying process. In this way, human resources are freed from the task, and the efficiency is improved in the meantime. However, there's still one disadvantage of this seemingly perfect system: upgrade cost. According to the official website of the Singapore Land Transport Authority, an IU costs driver \$150, let alone the cost to install all the RFID equipment required. In developing countries/regions, there are thousands of traditional parking lots. With the high-cost level, they

can't upgrade promptly. The system needs to satisfy the following requirements to solve this problem:1. Require few or no other hardware deployment;2. the communication between drivers and the system has to be fast and accurate.

Under the constraints mentioned above, I propose a smart parking access control system based on mobile Audio Content Recognition(ACR). In our system, I use Near Ultrasound Audio Tag algorithm to make it possible for drivers to pay the toll only using their mobile phones. During the payment process, the mobile-end application generates a near-ultrasound audio signal based on the user's profile information. Then the user can play it to the receiving device possessed by charging employees, which can be another mobile phone. After the receiving-end, analyze the signal and decode it into the tag number and send it to the server to check the validity of the user. When the server approves the transaction, the driver is good to go. Due to the utilization of the near-ultrasound frequency channel, the loud environment noises of the parking lot shall not influence the performance of the system. After the implementation of the system, I conduct several experiments under different sets of environmental parameters and compare them to existing middle-range communication techniques

5.2 System Architecture

In this section, we present the detail of the workflow and operation modes of the Parking Loud application. The user profile data is used to generate the audio tag signal for communication between the sender and the receiver. All the data and communication records generated during this process is stored. The operation mode decides which storing method to be used.

5.2.1 Workflow

This application is designed to help parking lots in developing countries or regions to upgrade at a low cost. Currently, many automatic parking lot control systems require some kinds of equipment. Like in Singapore [9], Radio Frequency Identification(RFID) [12] emitters and receivers to form the parking lot access control system. The RFID antenna is rather expensive, let alone every car owner needs to install an In-vehicle Unit(IU) by paying \$150 to the Singapore Land Transport Authority(LTA). These costs take a heavy toll both on the drivers' side and the managers' side. Hence I come up with a system whose architecture is shown in Figure 5.2. This system is designed to work in a poor environment. The core components of this system are sender, receiver, and storage devices, both of which can be simply mobile phones.

- *Sender*

The sender takes charge of transforming the user's profile data(including user basic information, vehicle information) into an audio tag signal. Then this signal is sent out.

- *Receiver*

The receiver keeps recording and analyzing the surrounding environment. Once the receiver detects a complete signal, it decomposes the signal to extract the necessary information to finish the transaction.

- *Storage*

The storage unit can be rather flexible in this system. The financial state and the network connection state of the assumption of developing countries and regions can be both impoverished. Hence, the server should not be a fixed component. Thanks to the rapid development in mobile devices,

data storage can be achieved even within a phone. This allows the system to work in two modes: online mode and offline mode. It can be toggled depending on the environmental parameters.

5.2.2 Operation Mode

The Parking Loud system is originally designed to help traditional parking lots, which mostly exist in developing countries/regions, upgrade. I want our system to be capable of working under poor environmental conditions, like bad network connection and lack of server resources. Hence, I implement 2 operation modes for the system: online mode and offline mode. The main difference between these 2 operation modes is the location where the data is stored. In online mode, all the data is stored in a server that takes charge of functionalities like authentication and user management. In contrast, these works are done by the receiver phone. By in offline mode to trade robustness for the much lower total cost.

5.3 Experiments

Parking Loud application can work well in a real parking lot environment to serve as an automatic parking lot control system with short paying time and low energy cost. In the real parking lot experiment, we choose three parking lots in NTU. They are located in different kinds of environments, different capacities, and different car traffic. We use Samsung Note 8 as the sender and Asus smartphone as the receiver. The experiment is conducted in 3 distances: 0.5m, 1.5m, 3.5m. Because it's a simulation to real application scenarios, these distances are closer to the real distance between the driver and the parking-lot access control system. The experiment results are shown in Figure 5.3.

Figure 5.3: Experiment Result of min/mean/max recognition time cost in Real NTU Parking Lot environment. The recognition time cost increases as the distance between the sender and the receiver increases.

From the result, we can observe that the Parking Lot application can work as an access control system in the parking lot environment with acceptable performance loss. Compared to the silent room performance experiment result in table 3.3, the recognition time increases averagely by 1 second. We think this is normal due to the noises produced by vehicle covers from low frequency to high frequency, which can have a significant impact on the system's functionality. In the living room experiment, the acoustic interference focus on the low-frequency part of the spectrogram(mainly 0-6k Hz) where human voices and TV sounds, such as instrumental music and object collision, reside.

In all the experiment settings, the 0.5m experiment has the best performance with an average mean recognition time of 1.7 seconds. As we observe in the parking lot, the distance between the driver and the barrier gate is around 0.5m to 1.5m. In this distance range, we can see that our Parking Lot application can finish the communication under 2 seconds, which outperforms traditional parking lot where managers or self-serving machines charge a parking fee. Interestingly, we also notice that in the parking lot A experiment, the mean recognition time seems abnormally larger than others. After listening to the audio samples, we found appearances of irregular high-frequency noises that can effectively disable the audio tag algorithm. With audio fingerprint as auxiliary recognition methods, our parking loud application manages to recognize the signal. The good performance of the application in the real parking lot strongly proves the robustness and stability of our application in a realistic environment. From that, we get 2 conclusions: 1)high-frequency noises are rare in the normal environment. Base on that, we can conclude that Parking Loud system can work most of the time properly. 2) Even with the interference of noises, the system still manages to recognize the signal in all experiment settings, which shows good robustness.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

The problem of understanding the information contained by acoustic signals can be solved by utilizing cognitive computing techniques and rapidly prevailing mobile devices. In this paper, I propose and implement 3 mobile applications and 2 algorithms to support the functions of them. We conduct a series of experiments to test the performance of the audio tag and audio fingerprint algorithms, and the results prove that they can recognize the input audio samples in a short period with good recognition accuracy. Based on them, I build 3 applications to explore the possibility of deploying them in the real world.

The Hey!Shake application enables users to interact with the TV programs and substantial product sellers instead of merely watching it. I build the system based on the audio tag and audio fingerprint algorithm. By integrating these 2 algorithms, the application is capable of knowing what multimedia contents the customers are watching right now. In this way, the content provider or video platform maintainer can provide much more precise and useful ads or product recommendation services.

The Parking Loud application makes it possible for parking lots of owners who can't afford the traditional automatic access control system to upgrade with merely two microphone-equipped devices. I propose and build a low-cost, easy-

to-use parking lot access control system. Given none of the previous works try to combine acoustic content recognition techniques with parking lot access control, we design an audio tag algorithm and build a system on top of it. The performance experiment results show that our system can work well in real environmental conditions with reasonable waiting time accuracy and low total cost. However, I notice that our system can still be improved. Like in real parking lot experiments, the recognition time becomes longer in 3.5m compared to the silent room experiment. The difference suggests that in a real environment, extra noises make energy decay a bigger problem.

Audio possesses a lot of hidden channel information that can be utilized to improve the quality of existing services and applications. With the development in mobile devices and machine learning techniques, it's possible for not only researchers but those software developers and people with a little research background to understand and to deploy audio-assist cognitive computing systems.

6.2 Future Works

Many different auxiliary experiments have been left for the future due to lack of time. Future work concerns some deeper analysis of particular mechanisms, new proposals to try different methods.

There are some ideas that I would have liked to try and further improve the performance of the algorithm. This thesis has mainly focused on the audio tag algorithm, audio fingerprint algorithm, and the ability of them to build efficient systems to solve real-world problems in developing countries/regions with poor economic foundations. Here are some ideas that can be tested in the future:

- The future research direction should be looking for more effective encod-

ing/decoding methods and more various application scenarios of utilizing hidden channel information. Currently, the audio tag algorithm and audio fingerprint algorithm are mostly in its prototype form. Though experiments are conducted to test their performance in several simulated environments, how they are adopted in real-world applications is still unclear to us.

The performance experiments described in Chapter.3 propose several potential improvements:

- Improve the effective range, robustness of the ACR algorithms
- Shorten the segment length of the audio tag algorithm without jeopardizing the recognition time.
- Propose new hashing and partitioning algorithm for audio fingerprint algorithm

It could also be interesting to explore the possibilities of the ACR algorithms since the application of them is limited. The noise attack and recording distortions have a significant influence on the recognition accuracy and time cost, which in turn confine the use of the two sample applications in the middle-range. More work is required to break that limitation. The use of consecutive near-ultra signal has some unchangeable flaws. It's possible to use only very short frequency peaks and use the pattern to identify different media contents.

- Besides the traditional audio algorithms, machine learning techniques can help to add more functionalities to the existing applications. Machine learning techniques currently are mainly targeting simple recognition tasks. How people can develop an automatic understanding of audio content is still beyond our imagination. But in recent years, the booming development

in multimodal machine learning has shown us a promising path to solve this problem since human brains are functioning in multimodal ways. In my opinion, the work presented in this thesis can be upgraded by applying multimodal techniques, and this is my main research focus in the future.

Bibliography

- [1] M. A. Alsalami and M. M. Al-Akaidi, “Digital audio watermarking: survey,” *School of Engineering and Technology, De Montfort University, UK*, 2003.
- [2] Y. Aytar, C. Vondrick, and A. Torralba, “Soundnet: Learning sound representations from unlabeled video,” in *Advances in Neural Information Processing Systems*, 2016, pp. 892–900.
- [3] h. SoungHound.
- [4] C. Van Loan, *Computational frameworks for the fast Fourier transform*. Siam, 1992, vol. 10.
- [5] T. Ganchev, N. Fakotakis, and G. Kokkinakis, “Comparative evaluation of various mfcc implementations on the speaker verification task,” in *Proceedings of the SPECOM*, vol. 1, no. 2005, 2005, pp. 191–194.
- [6] M. Müller, *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer-Verlag, 2007.
- [7] V. Tyagi and C. Wellekens, “On desensitizing the mel-cepstrum to spurious spectral components for robust speech recognition,” in *Proceedings.(ICASSP’05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 1. IEEE, 2005, pp. I–529.
- [8] D. F. Nelson, “Parking lot exit control means,” Jul. 18 1978, uS Patent 4,101,235.
- [9] T. Obata, H. Ono, Y. Miyazaki, and M. Ando, “Electronic parking system for singapore,” *Mitsubishi Heavy Industries Technical Review*, vol. 40, no. 3, 2003.
- [10] G. Ostojic, S. Stankovski, M. Lazarevic, and V. Jovanovic, “Implementation of rfid technology in parking lot access control system,” in *2007 1st Annual RFID Eurasia*. IEEE, 2007, pp. 1–5.
- [11] S. Ka, T. H. Kim, J. Y. Ha, S. H. Lim, S. C. Shin, J. W. Choi, C. Kwak, and S. Choi, “Near-ultrasound communication for tv’s 2nd screen services,”

- in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 2016, pp. 42–54.
- [12] K. Finkenzeller and R. Waddington, *RFID Handbook: Radio-frequency identification fundamentals and applications*. Wiley New York, 1999.
 - [13] S. A. Craver, M. Wu, and B. Liu, “What can we reasonably expect from watermarks?” in *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*. IEEE, 2001, pp. 223–226.
 - [14] J. Dittmann, A. Mukherjee, and M. Steinebach, “Media-independent watermarking classification and the need for combining digital video and audio watermarking for media authentication,” in *Information Technology: Coding and Computing, 2000. Proceedings. International Conference on*. IEEE, 2000, pp. 62–67.
 - [15] M. D. Swanson, B. Zhu, and A. H. Tewfik, “Current state of the art, challenges and future directions for audio watermarking,” in *Multimedia Computing and Systems, 1999. IEEE International Conference on*, vol. 1. IEEE, 1999, pp. 19–24.
 - [16] S. D. M. Initiative, “Sdmi - home, <http://www.sdmi.org/>,” 2002. [Online]. Available: <http://www.sdmi.org/>
 - [17] J. Vanderkooy and S. P. Lipshitz, “Resolution below the least significant bit in digital systems with dither,” *Journal of the Audio Engineering Society*, vol. 32, no. 3, pp. 106–113, 1984.
 - [18] D. Gruhl, A. Lu, and W. Bender, “Echo hiding,” in *International Workshop on Information Hiding*. Springer, 1996, pp. 295–315.
 - [19] K. Steiglitz, I. Kamal, and A. Watson, “Embedding computation in one-dimensional automata by phase coding solitons,” *IEEE Transactions on Computers*, no. 2, pp. 138–145, 1988.
 - [20] R. C. Dixon, *Spread spectrum systems: with commercial applications*. Wiley New York, 1994, vol. 994.
 - [21] I.-K. Yeo and H. J. Kim, “Modified patchwork algorithm: A novel audio watermarking scheme,” *IEEE Transactions on speech and audio processing*, vol. 11, no. 4, pp. 381–386, 2003.

- [22] P. Bassia and I. Pitas, "Robust audio watermarking in the time domain," in *Proc. EUSIPCO*, vol. 98. Citeseer, 1998, pp. 25–28.
- [23] P. Bassia, I. Pitas, and N. Nikolaidis, "Robust audio watermarking in the time domain," *IEEE Transactions on multimedia*, vol. 3, no. 2, pp. 232–241, 2001.
- [24] P. Welch, "The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Transactions on audio and electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.
- [25] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.
- [26] M. J. Shensa, "The discrete wavelet transform: wedding the a trous and mallat algorithms," *IEEE Transactions on signal processing*, vol. 40, no. 10, pp. 2464–2482, 1992.
- [27] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische mathematik*, vol. 14, no. 5, pp. 403–420, 1970.
- [28] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE transactions on image processing*, vol. 6, no. 12, pp. 1673–1687, 1997.
- [29] M. Arnold, "Audio watermarking: Features, applications and algorithms," in *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No. 00TH8532)*, vol. 2. IEEE, 2000, pp. 1013–1016.
- [30] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM systems journal*, vol. 35, no. 3.4, pp. 313–336, 1996.
- [31] T. Painter and A. Spanias, "Perceptual coding of digital audio," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 451–515, 2000.
- [32] D. G. Hong, S. H. Park, and J. Shin, "A public key audio watermarking using patchwork algorithm," *Proceedings of ITCCSCC*, pp. 160–163, 2002.
- [33] Shazam. Shazam music recognition service, <http://www.shazam.com/>.
- [34] h. SoungHound.

- [35] S. Baluja and M. Covell, “Content fingerprinting using wavelets,” 2006.
- [36] —, “Audio fingerprinting: Combining computer vision & data stream processing,” in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP’07*, vol. 2. IEEE, 2007, pp. II–213.
- [37] M. Fink, M. Covell, and S. Baluja, “Social-and interactive-television applications based on real-time ambient-audio identification,” in *Proceedings of EuroITV*. Citeseer, 2006, pp. 138–146.
- [38] A. Wang *et al.*, “An industrial strength audio search algorithm.” in *Ismir*, vol. 2003. Washington, DC, 2003, pp. 7–13.
- [39] Y. Ke, D. Hoiem, and R. Sukthankar, “Computer vision for music identification,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1. IEEE, 2005, pp. 597–604.
- [40] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [41] M. Covell and S. Baluja, “Known-audio detection using waveprint: Spectrogram fingerprinting by wavelet hashing,” in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP’07*, vol. 1. IEEE, 2007, pp. I–237.
- [42] B. Logan *et al.*, “Mel frequency cepstral coefficients for music modeling.” in *ISMIR*, vol. 270, 2000, pp. 1–11.
- [43] K. J. Piczak, “Environmental sound classification with convolutional neural networks,” in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.
- [44] I. McLoughlin, H. Zhang, Z. Xie, Y. Song, and W. Xiao, “Robust sound event classification using deep neural networks,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 540–552, 2015.
- [45] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, “Unsupervised feature

- learning for audio classification using convolutional deep belief networks,” in *Advances in neural information processing systems*, 2009, pp. 1096–1104.
- [46] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [47] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, “Cnn architectures for large-scale audio classification,” in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on.* IEEE, 2017, pp. 131–135.
- [48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [49] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [50] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [51] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [52] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on.* IEEE, 2017, pp. 776–780.
- [53] W. W. Peterson and D. T. Brown, “Cyclic codes for error detection,” *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.
- [54] S. A. Gelfand, *Hearing: An introduction to psychological and physiological acoustics.* CRC Press, 2017.

- [55] D. Ellis, “Landmark-based audio fingerprinting,” <https://github.com/dpwe/audfprint>, 2018.
- [56] D. C. Shoup, “Cruising for parking,” *Transport Policy*, vol. 13, no. 6, pp. 479–486, 2006.
- [57] D. Shoup, *The High Cost of Free Parking: Updated Edition*. Routledge, 2017.