A TEMPO-BASED MUSIC SEARCH ENGINE WITH MULTIMODAL QUERY

YI YU

B.Sc. OF ENGINEERING

TSINGHUA UNIVERSITY

2009

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2011

Abstract

This thesis presents TMSE: a novel Tempo-sensitive Music Search Engine with multimodal inputs for wellness and therapeutic applications. TMSE integrates six different interaction modes, Query-by-Number, Query-by-Sliding, Query-by-Example, Query-by-Tapping, Query-by-Clapping, and Query-by-Walking, into one single interface for narrowing the intention gap when a user searches for music by tempo. Our preliminary evaluation results indicate that multimodal inputs of TMSE enable users to formulate tempo related queries more easily in comparison with existing music search engines.

Acknowledgement

This thesis would not have been possible without the support of many people.

Greatest gratitude to my supervisor, Dr. Wang, who offered valuable support and guidance since I started my study in School of Computing.

I would like to thank all my friends for their suggestions and help.

I am deeply grateful to my beloved families, for their consistent support and endless love.

I would like to thank Dr. Davis, Dr. Dixon, Dr. Ellis and Dr. Klapuri for making their source code available.

Without the support of those people, I would not be able to finish this thesis. Thank you very much!

Contents

List of Publications

A Music Search Engine for Therapeutic Gait Training,
Z. Li, Q. Xiang, J. Hockman, J. Yang, Y.Yi, I. Fujinaga, and Y. Wang. ACM Multimedia International Conference (ACM MM), 25-29th October 2010, Firenze, Italy.

A Tempo-Sensitive Music Search Engine With Multimodal Inputs,
Y. Yi, Y. Zhou, and Y. Wang. ACM Multimedia International Conference (ACM MM) Workshop on MIRUM 2011.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Tempo is a basic characteristic of music. The act of tapping one's foot in time to music is an intuitive and often unconscious human response [Dix01]. Content-based tempo analysis is important for certain applications. For example, music therapists use songs with particular tempi to assist the Parkinson's disease patients with gait training. This method, also known as rhythmic auditory stimulation (RAS) [TMR+96], has been shown to be effective in helping these patients achieve better motor performance. Music tempo can also facilitate exercises when people listen to music and run according the music beats. It motivates them to run and makes them feel less tired [OFM06].

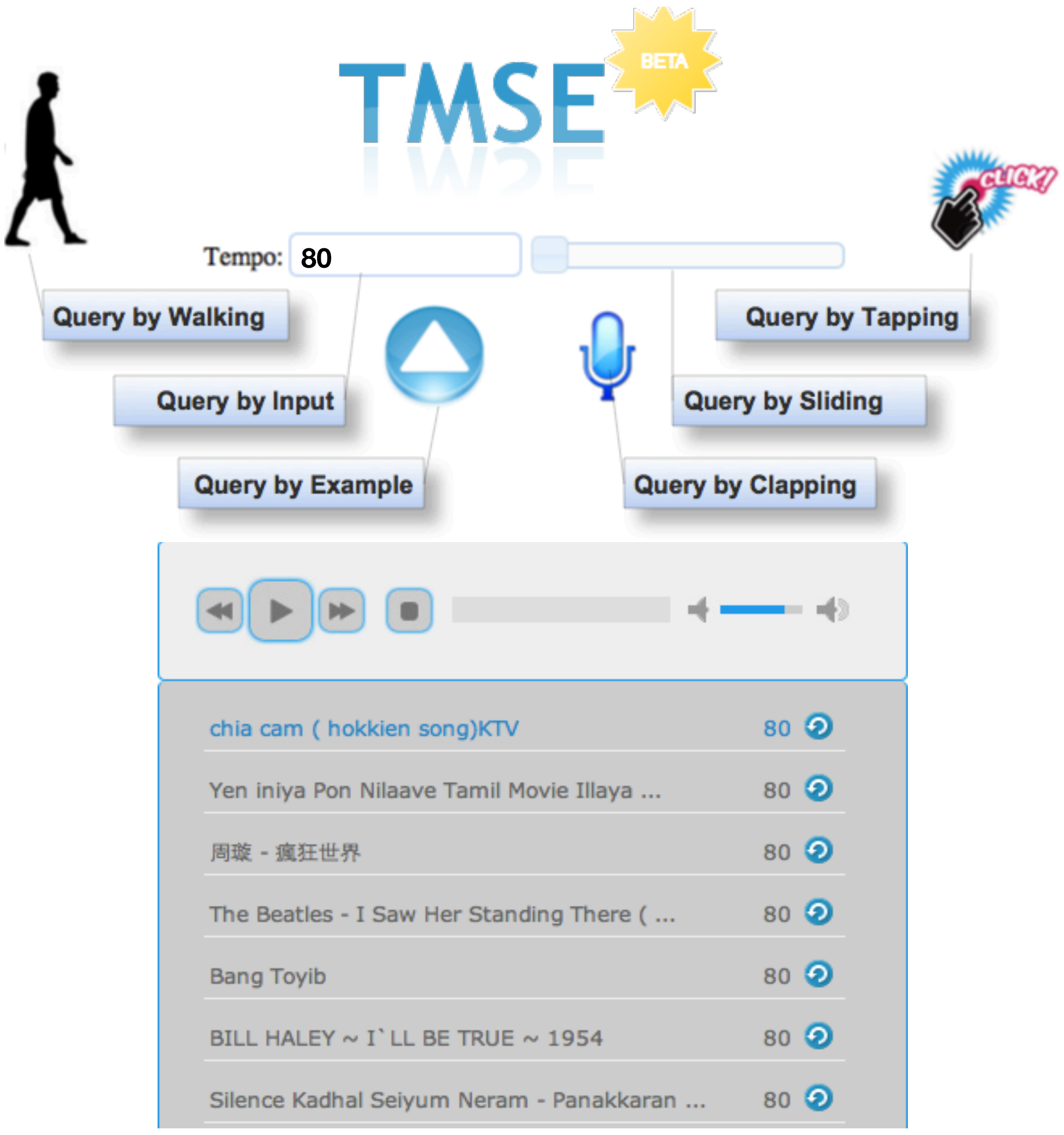
In above scenarios, a well-designed music search engine for tempo will help users to achieve the goal: users can search for a list of songs based on the tempo information. However, the search box in a traditional search engine constrains the way to express the music tempo in a query. Although it is easier for users with music background (e.g. trained musicians or music therapists) to input a number as BPM (*beats-per-minutes*) value to accomplish their queries, it makes little sense to ordinary users who do not understand the mere number of BPM. Most of ordinary users can hardly express it in the form of a rigid number, even when they have their desired tempo to search. The text input mode hampers users' expression of the music tempo that occurs in their mind, which forms so-called *Intention Gap*. This motivates us to develop a tempo-sensitive music search engine.

Some critical questions about tempo-sensitive search engine include:

1. How to estimate the tempo of a song?
2. How to design a user interface with more choices that enables users to express their intention easily?
3. How to design a user interface that allows users to verify the tempo values returned by the search engine?

For the question 1 itself, we adopted 4 existing beat tracking methods to estimate tempo followed by evaluations. We chose the best tempo estimator in our system implementation. This is a part of an ongoing project [LXH+10].

The focus of this thesis is to address questions 2 and 3. For users to express music tempo with more choices (question 2), we provide users a tempo-sensitive music search engine with multimodal inputs to express their intention in tempo. The multimodal inputs include (see Figure 1.1):



chia cam (hokkien song)KTV 80
Yen iniya Pon Nilaave Tamil Movie Illaya ... 80
周璇 - 瘋狂世界 80
The Beatles - I Saw Her Standing There (... 80
Bang Toyib 80
BILL HALEY ~ I`LL BE TRUE ~ 1954 80
Silence Kadhal Seiyum Neram - Panakkaran ... 80

Figure 1.1: TMSE User Interface

- **Query-by-Number**: A user inputs a tempo value in BPM; TMSE returns a list of songs which are close to the tempo value.

- **Query-by-Tapping**: A user taps the mouse; TMSE returns songs with tempo values which are close to the user's tapping speed.
- **Query-by-Example**: A user uploads a song; TMSE returns songs with similar tempi of the example song.
- **Query-by-Clapping**: A user claps his/her hands; TMSE returns songs matched to the speed of clapping.
- **Query-by-Sliding**: A user retrieves songs in different tempi by sliding a slider for tempo. This is designed for users without music background, so that they could have a sense of quick and slow tempo.
- **Query-by-Walking**: When a user carries an iPhone, TMSE detects the users walking speed and returns songs close to the walking speed.

In order to verify the search results from TMSE (question 3), a user can listen to the results while clicking the mouse. The estimated music tempo and the clicking tempo will be displayed in the interface side by side thus allowing an easy comparison. To further enhance the user experience, we provide users a button near the song name, which can modify the tempo of the song without changing its pitch.

The research contributions of this thesis are as follows:

- We have developed a tempo-sensitive music search engine with multimodal inputs, called TMSE, to fulfill users' search requirements of tempo, which is typically lacked in the existing music search engines.

- We have developed an eyes-free application on iPhone, called iTap, to annotate music tempo in order to collect ground truth for our tempo-sensitive search engine.
- We have conducted a preliminary usability study of the proposed prototype. We have shown that music tempo searchability is an important aspect missing in most existing music search engines.

1.2 Organization of the Thesis

The organization of the thesis is as follow: Chapter 2 provides the literature review of all related research topics of this thesis. Chapter 3 describes the *iTap* with its implementation details. iTap is an eyes-free mobile framework for collecting the tempo annotation as the ground truth. Chapter 4 describes the TMSE in details, especially its six different query inputs. Chapter 5 describes the evaluation of the system. An evaluation on 4 beat tracking algorithms is presented, as well as a preliminary user study. Chapter 6 describes the future work of the thesis, including the future work for both iTap and TMSE. Chapter 7 concludes the thesis by summarizing its main research contributions.

Chapter 2

Related Work

2.1 Query Inputs in Music Information Retrieval

According to [TWV05], most of the existing music information retrieval systems can perform either very general or very specific retrieval tasks. Recently, researchers in music information retrieval have been trying to use multiple novel inputs to reduce the users' intention gap. In our system, we use the combination of multiple query inputs to reduce the intention gap for users.

We will review some existing popular query inputs in the following subsections. All these query inputs are used for improving the user experience in Music Information Retrieval (MIR). However, none of these is capable of searching for music according to tempo. In our TMSE, Query-by-Number, Query-by-Sliding, Query-by-Example, Query-by-Tapping, Query-by-Clapping and Query-by-Walking are integrated into one single user interface to facilitate users to express tempo query.

2.1.1 Query-by-Example

Tzanetakis et al. [TEC02] develop a new query interface for Query-by-Example. In the system, algorithms are used to extract the content information from the example audio signals, and the extracted information is used to configure the graphical interfaces for browsing and retrieval.

Harb et al. [HC03] propose a set of similarity measures based on the distance between statistic distributions of audio spectrum. A Query-by-Example system relying on the presented similarity measures is evaluated.

Tsai et al. [TYW05] propose a Query-by-Example framework, which takes a fragment of a song as the input, and returns songs similar to the query as the output. The returned songs is similar to input in terms of the main melody. This method is based on the similarity between the note sequences.

Shazam [Wan06] is a successful and popular Query-by-Example iPhone application. Users can record a part of a song from the environment as the query, and the system then searches for the songs which could be the query song from the server.

2.1.2 Query-by-Humming

CubyHum is a Query-by-Humming system developed by Pauws [Pau02]. CubyHum identifies a song from a given sung input. Algorithms for pitch detection, note onset detection, quantitation, melody encoding and approximate pattern matching are implemented in the CubyHum system.

Lu [LYZ01] et al. propose a Query-by-Humming system, which uses a new melody representation and a new hierarchy matching method in order to adapt to users' humming habits.

Midomi [Web] is a popular iPhone application for Query-by-Humming. In Midomi, a user can sing into the iPhone's microphone, and the system can search for a song that is similar to the user's humming.

2.1.3 Query-by-Tapping

Jang et al. [JLY01] propose a Query-by-Tapping system. The system takes the user input in the form of tapping on the microphone, and then the extracted duration of notes is used to retrieve the intended song from the database.

Hanna et al. [HR09] propose a retrieval system using symbolic structural queries. The system allows users to tap or clap into microphone to search for music. The search is based on the rhythmic pattern of the melody.

2.2 Beat Tracking Algorithms

Music beats correspond to the time when a human listener would tap his foot. *Beat Tracking* is a technique trying to track every beat in a song. Beat Tracking is not only essential for computational modeling of music, but also fundamental for MIR (Music Information Retrieval).

Much research is done on different beat tracking algorithms.

Early approaches [AD90, Ros92, Lar95] for beat tracking process symbolic data rather than audio signals, due to lacking of computational resources in note onset detection.

For beat tracking directly from audio data, many approaches rely on the automatic extraction of the note onsets.

Dixon's method, *BeatRoot* [Dix01, Dix06, Dix07], employs multi-agent method based on inter-onset interval (IOI). An onset detection algorithm is performed to get all the onsets. Then BeatRoot processes the sequence of note onsets within a multi-agent system. Likely tempo hypotheses are derived from clustering inter-onset-intervals (IOIs). They are used to form multiple beat agents. Finally one of these best agent is selected, because of its best prediction of beat locations. BeatRoot is designed to track beats in expressively performed music.

Goto's method [Got01] is also based on agent method. More than tracking the beats (1/4 note level) only, Goto's approach also tracks 1/2 note level and whole note level. Onset analysis is performed across seven parallel sub-bands, where spectral models are used to extract snare and base drum events. Then beats and higher-level musical structures are inferred from chord changes and pre-define rhythmic patterns. Goto's system operates accurately and in real-time when it deals with a steady tempo and a 4/4 time signature.

Hainsworth's system [HH03] performs onset detection using two distinct modals, and combines the results from both to give a single sequence of onsets. Then particle filters are used within a statistical framework to track the beats.

In contrast to the previous onset-based approaches, there are other approaches.

Instead of detecting onsets only, Scheirer [Sch98] emphasizes the onset locations using a psycho-acoustically motivated amplitude envelope signals. This forms the front-end of the beat tracker. Six octave-spaced envelope signals are passed through a parallel bank of tuned comb filter resonators, which represent tempo hypotheses over the range of 60-180 BPM. Then a real-time system infers the tempo and predicts the phase of the beats simultaneously using phase-lock.

Klapuri's method [KEA06] expands upon Scheirer's amplitude envelope and comb filter model. This approach analyzes the basic patterns of music meters: tatum, tactus (beat), and measure. A technique measuring the degree of musical accent acts as the initial time-frequency analysis, followed by a bank of comb filter resonators. Finally a probabilistic model is used to perform joint estimation of the tatum, tactus, and measure pulses. The Klapuri's approach is considered as the state-of-the-art approach in some literatures [GKD$^+$06].

Davies and Plumbley's approach [DP07] uses an onset function instead of note onsets as the intermediate representation of audio. Auto-correlation is used on the onset function to induct tempo, followed by a two-state models: the algorithm stays in the first state to track beats in a stable tempo, and goes into the second state when it finds tempo changes.

Ellis [Ell07] formalizes the beat tracking problem using dynamic programming algorithm. The algorithm includes two parts: 1) tempo estimation of a song; 2) using tempo as the constrain of the dynamic programming, and find the best-scoring set of beat time. This algorithm relies on a constrained function of tempo, and the advantage of dynamic programming make it efficient to solve the problem. However, a stable tempo is the premise for the algorithm working correctly. So this approach works poorly when dealing with tempo changes.

Due to the availability of source code, in this thesis we evaluate 4 algorithms: the Dixon's, Klapuri's, Davies' and Ellis'. The ones with best performance are implemented in TMSE.

2.3 Eyes-free Application

In our system, we need a set of *ground truth* to evaluate different beat tracking algorithms, so that we can choose the best beat tracking algorithm. In order to get these ground truth, i.e., human annotated music tempo for many music pieces, an annotation tool is needed Traditional approaches require many visual attention, which is not necessary. We develop an eyes-free annotation tool, iTap, to help people annotate music without much visual attention.

Here is a literature survey on eyes-free applications.

Some of them are mobile-based auditory feedback UIs, such as auditory icon from Graver and Smith [GS91], earcons from [BWE93], and earPod from Zhao [ZDC$^+$07].

Kamel et al. [KL02] invents an effective drawing tool for the blind. This method transforms a mouse-based graphical user interface into a navigable, grid-based auditory interface.

BlindSight's work [LBH08] helps users to access mobile phones in an eyes-free environment. This method replaces the visual interface with one based on auditory feedback, and users can interact using the phone keypad without looking at the screen.

Bach et al. [BEC$^+$07] develop a two-way speech-to-speech translation system, using a completely eyes-free/hands-free user interface.

However, none of them are focused on developing eyes-free tools for music tempo annotation. In this thesis, iTap, as an iPhone app is developed to collect users' tempo annotation in an eyes-free environment.

2.4 Games-With-A-Purpose (GWAP)

We used iTap to collect tempo annotation on our music collections. In the process of annotation, we found that annotating large music collections is a boring and tiring task, even in an eyes-free environment. This motivates us to extend the iTap as Games-With-A-Purpose (GWAP) in our future work. GWAP makes the annotation task fun and easy enough, and it motivates people to annotate as many music pieces as possible. Here is a literature survey of GWAP, and later a detailed future research plan for GWAP research is described in Section 6.1.

The term *GWAP* can be explained as the following: people playing GWAP to have fun, and data generated as a side effect of the game play also solves computational problems and trains AI algorithms. There are three basic design patterns can be observed from the existing GWAP researches [VAD08]:

Output-agreement. ESP Game [VAD04, VA06] , a.k.a. the Google Image Labeler [1], is a GWAP in which people provide meaningful, accurate labels for images on the Web as a side effect of playing the game. For example, an image of a man and a dog is labeled dog, man, and pet. Two users are assigned to play this game simultaneously, and they can get scores only when they input same labels. Another example in this category is Matchin [HVA09], which collects users' "preferred pictures" between two pictures.

Inversion-problem. Peekaboom [VALB06], Phetch [vAGKB07], and Verbosity [VAKB06] can be classified into this category. Peekaboom is used to locates objects within images, Phetch is used to annotate images with descriptive paragraph, and Verbosity collects commonsense facts in order to train reasoning algorithms. In the all above three games, two players are involved. One of them acts as the describer and the other acts as the guesser. Partners are successful only

[1]http://images.google.com/imagelabeler/

when the describer provides enough outputs for the guesser to guess the original input.

Input-agreement. TagATune [LVADC03], can be a good example in this category. TagATune labels music pieces with tags of personal feelings e.g. “dark”, “driving” or “creepy”. Two users produce output for their own inputs, and they can see is the others’ output. In the end of each session both users have to agree whether they are given the same music piece.

In each of the games mentioned above, people play not because they are personally interested in solving an instance of a computational problem but because they wish to be entertained [VAD08].

As all GWAP listed above require visual attention, designing eyes-free GWAP is still possible with music applications. We will explore the potential of extending iTap as GWAP in Section 6.1.

Chapter 3

iTap

3.1 Introduction

It is critical to have a manually annotated music database for developing a tempo-sensitive music search engine.

We need to evaluate different beat tracking algorithms to select the best-performance algorithm. If we only evaluate beat tracking algorithms on some computer-generated music pieces with known tempi, which will cause some disadvantages and consequences:

1. The music we intent to use in our search engine are *real-life* music. The real-life music means that the music piece usually contains multiple sound tracks, consisting of different acoustical music instruments. If we want to

evaluate algorithms, we need to evaluate them in our target music and in a real scenario.

2. The music we intent to use are *expressive* music. The term expressive means, although the music is in a relatively stable tempo, the timing is still slightly different from beat to beat, due to the nature of human performance. Generally speaking, all the state-of-the-art algorithms perform well on the computer-generated music according to the literature. We would like to see how these algorithms perform on expressive music.

On the other hand, if we could build a music database with annotated tempi by human, this music database can serve as the *ground truth* for our algorithm evaluation, and the evaluation results on this ground truth are reliable in our search engine system. That is why we prefer manually annotated tempi rather then computer-generated music.

It is possible to do the annotation using a PC with keyboard or mouse [Bro05]. However, a PC-based annotation tool forces the subject to sit in front of a computer to accomplish the annotation task with full visual attention, which makes the annotation task boring and stressful.

On the other hand, tapping along with music is an unconscious human response [Dix01]. Therefore full visual attention is not necessary throughout the whole tempo annotation process. This motivates us to develop an eyes-free annotation tool, *iTap*, on mobile devices. iTap provides the following benefits:

1. An eyes-free tempo annotation tool allows the subjects to perform the anno-

tation tasks anywhere and anytime. For example, the subjects can annotate tempo while walking or commuting in a bus or subway with little visual attention.

2. We intend to extend the iTap to become GWAP (Games-With-A-Purpose) as future work, in order to motivate more subjects to annotate our music database.

3.2 System Architecture

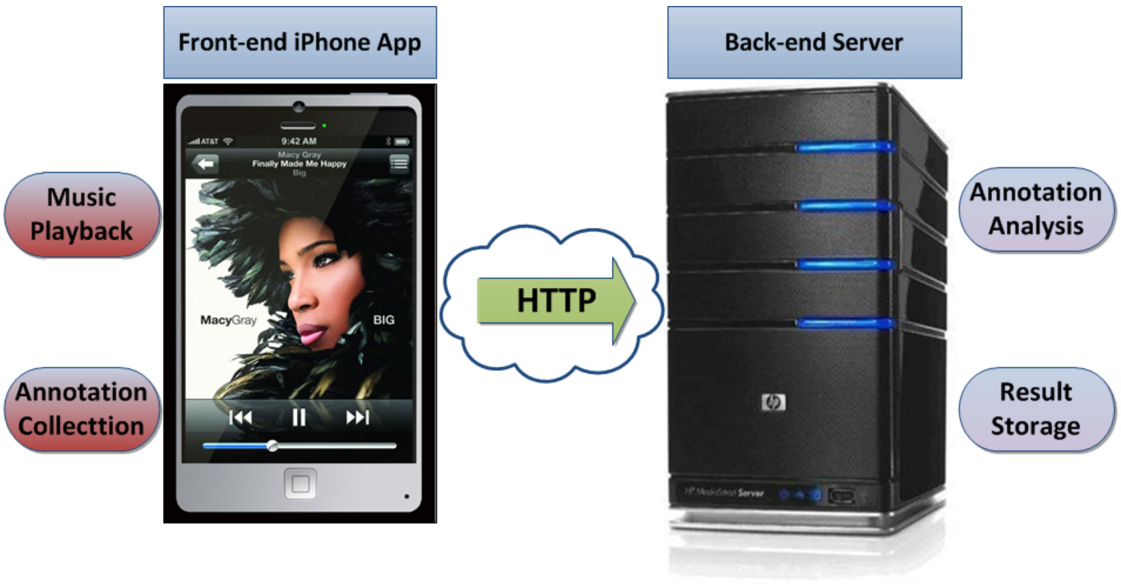


Figure 3.1: The architecture of iTap

Figure 3.1 depicts the architecture of the iTap system, omitting the user interface component.

iTap uses the classic client-server architecture. Under client-server architecture, a user can annotate music piece in any place any time, as long as s/he wants to. When the Internet access is available, the annotation results are uploaded onto our server, processing automatically, and storing in the database permanently.

This front-end, iTap, is written by Object-C programming language, as an iPhone App. It plays the most important role in the annotation system: music playback as well as annotation collection. The iTap can be installed on an iPod Touch or an iPhone. iTap playbacks music tracks, then a user touches on the screen when he feel he can follow the tempo. At the same time iTap records every tap of the user. After annotating from a user, these data will be sent to server through a HTTP request when the Internet is accessible. The whole annotation process needs very little users' visual attention, and the main tasks for the user are listening and tapping.

Server is written by PHP programming language, running on a Linux server environment. Server part is mainly designed for process the annotation data, and store users' annotation permanently in the database. Firstly the server will accept the HTTP request and exact all the raw data. And then raw data will be put into a table of a MySQL database for future use. From the raw data table, server can calculate final tempi values, using different algorithm. These tempi values will be stored in other separate database tables.

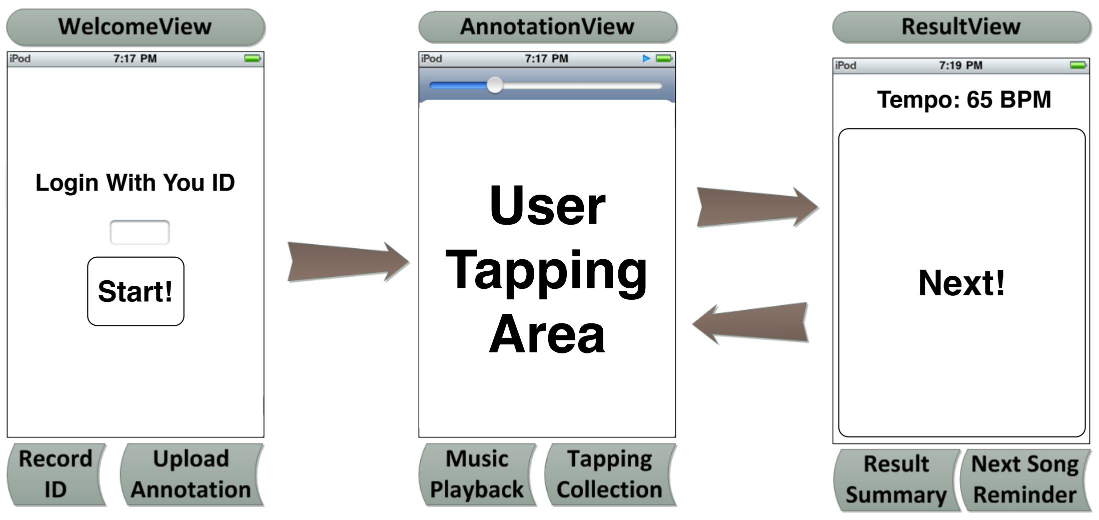


Figure 3.2: The Interface of iTap

3.3 Front-end: iTap

We chose iPhone platform as the front-end for the following reasons: fast-growing users, beautiful and easy user interface, smoothest touch screens, good network supporting and a relative mature SDK on mobile devices.

Figure 3.2 depicts the user interface and workflow of iTap.

The workflow of iTap is described as follow. Once users submit their IDs, the system generates a music playlist from which they can listen to a 30-second music excerpt randomly chosen from each song. Users start tapping on the screen as long as they can follow the beats of the music while listening. After tapping one piece of music, iTap reminds users to go on to the next one. The tapping area is large enough so that users can tap on it without looking at it. If users want to skip a song, they just slide to the right on the tapping area; if users want to re-play a song, they just slide to the left. Throughout the annotation process, very little visual attention is needed. Users only need to wear earphones and listen to the music to tap, even when they are walking or commuting on a bus or subway.

For the user interface part, the *WelcomeView* is provided as the first view for users. Users need an id to start the annotation process. The *AnnotationView* is used to collect tapping from users. Users can listen to the music and annotate in this view; users also can skip a song or re-play a strong if he could not follow the tempo of a song. The *ResultView* is designed to allow users to relax between songs, and it also provides the annotation results of the last song.

The following subsections explain the functions of each view in details.

3.3.1 WelcomeView

WelcomeView(Figure 3.2: WelcomeView) is the entry of iTap, as well as the result collection view. Normally users only need to meet this view once at the beginning of their annotation. At the beginning, WelcomeView asks for a username as the AnnotatorID, which will be stored in the database associated with the tapping results indicating who annotate these data. User identification will be helpful when doing some post-filtering. The user can press the Start button to enter AnnotationView starting his/her annotation. After all the users annotation, last annotation results will be read from a temporary file and sent out through a HTTP request, and the temporary file will not be deleted once the uploading is successful.

3.3.2 AnnotationView

AnnotationView (Figure 3.2: AnnotationView) is used for collecting tapping from users, and is the most important view.

When users enter this view, the iTap will randomly choose a 30 seconds clip from a song to playback, e.g. 0s-30s or 150s- 180s, and this clip will be played repeatedly. Users are asked to tap on the screen when they can follow the tempo, and the absolute timestamps of every tapping will be recorded. If a user makes some mistakes, s/he can to slide to the left to "Replay": star annotating this piece again; If a user cannot follow the tempo of a particular song, s/he can simply slide to the right to "Skip": skip this music piece and start annotating next piece.

Because the touch screens of iPhones or iPod Touches are large enough, so that most of the time users only need to listen and tap, and little sight is needed. In fact, in the real use of the iTap, users annotated music when they are walking, taking a bus and even reading books most of the time.

After 15 times of tapping, the program will enter SummaryView automatically, and annotation results of this song will be written into a temporary file locally.

3.3.3 SummaryView

SummaryView (Figure 3.2: SummaryView) offers users a view to have a short break between two songs. The tempo of the last song user just annotated will be display in the center of the screen.

If a user has completed annotating all the songs, SummaryView will thank the user and inform her/him to quit the program.

3.4 Back-end: Server

Server program is written by PHP as a web service running on a Linux server. The web service receives HTTP requests from iTap, unpacks the data, and tempi are calculated using different methods. Afterwards, the program will connect to a MySQL database, and store all the results into different database tables.

The first table is for the raw data of tapping results. The timestamps of every tapping are stored in this table.

The second table is used for storing tempo calculated by mean method. The tempo is calculated from the mean value of the inter-tapping intervals (ITI), to be precise:

$$Tempo_{mean} = \frac{60}{\frac{1}{n} * \sum_{1}^{n} ITI_i}$$

And the third one is use for storing tempo calculating by median method. The tempo is calculated from the median value of the inter-tapping intervals (ITI) to be precise:

$$Tempo_{median} = \frac{60}{Median(ITI_i)}$$

All these data will be associated with SongName, AnnotatorID, Segment and

StartAnnotatingTime in the database table.

We use the tempi derived from the median values as the ground truth. We chose median instead of mean for the following reasons:

The mean is calculated by adding together all the values, and then dividing them by the number of values you have. As long as the data is symmetrically distributed this is fine - but the mean can still be thrown right out by a few extreme values (outliers), and if the data is not symmetrical (i.e. skewed) it can be downright misleading.

The median, on the other hand, really is the middle value. 50% of values are above it, and 50% below it. So when the data is not symmetrical, this is the form of 'average' that gives a better idea of any general tendency in the data.

As a summary, mean is affected largely by some outliers, and to the contrary median performs quite robust. In our application scenarios, users tap stably most of the time, but could be wrong far away in some taps (outliers). It is much more suitable for choosing median values to calculate tempi.

3.5 Annotation Process

Research shows that even lack of professional musical training, most people can tap in time with music, although trained musicians can follow the tempo more quickly and tap in time with music more accurately than non-musicians [DPB00].

Two amateur musicians, both play guitars for more than ten years, were hired to finish the annotation tasks. They were asked to use iTap to annotate the whole music dataset. The annotation results will be kept only when the results from both musicians matched.

We collect our music dataset directly from YouTube. The music dataset consists of around 800 songs, including: 200 Chinese songs, 200 Indian songs, 200 Malayan songs and 200 Western. We collect songs in this way because of the future clinic usage in Singapore, because the top three majorities of Singaporean population are: Chinese, Indian and Malayan.

Two amateur musicians finished their annotation tasks independently using their spare time. They annotated most of the songs when they were walking or commuting on a bus or subway. 560 out of 780 of their final annotation results are matched and kept as the ground truth.

Chapter 4

TMSE: A Tempo-based Music Search Engine

4.1 Introduction

TMSE is a tempo-based music search engine, with multi-modal inputs. TMSE is designed to reduce the intention gap, which is a gap between users' search intent and the queries, because of the incapability of key-word queries to express users intends.

Within TMSE, users can enjoy multi-modal inputs to query music in different tempo. These inputs include: Query-by-Number, Query-by-Sliding, Query-by-Tapping, Query-by-Clapping, Query-by-Example, and Query-by-Walking.

Here is the reason for choosing these query inputs.

1. Choosing Query-by-Number is because it is the most traditional query method.
2. Choosing Query-by-Sliding is because it can add the sense of fast/slow above Query-by-Number, and we would like to know whether it is effective.
3. Choosing Query-by-Tapping and Query-by-Clapping is because they can both act as natural response of human beings towards music tempo, and Query-by-Clapping is even more natural and interesting than Query-by-Tapping.
4. Choosing Query-by-Example is because it is used widely and effectively in other MIR applications.
5. Choosing Query-by-Walking is because its potential clinical use in the future, and also it is a challenging research problem.

An initial version of our tempo-sensitive music search engine was published in [LXH+10], which enables music therapists to search for music using Query-by-Number only. And the main idea of this thesis has been just published in [YZW11], focusing on the multi-modal inputs and its effectiveness in reducing intention gap.

4.2 System Architecture

As shown in Figure 1.1, we have designed the user interface such that users can easily choose a query mode among the six and receive results as a playlist containing songs with similar tempi.

For the implementation details: TMSE runs under Linux/Unix OS. The UI part of TMSE is developed in HTML5, CSS and AJAX, so that users can have the same user experience under all mainstream web browsers: IE 9, Chrome, and Firefox.

Two servers are used for TMSE: a web server and a media server.

The web server is web.py [wAOSWFiP], an open-source web framework based on Python programming language. Web.py deals with all the HTTP requests.

The media server is Red5 [Red] An open-source project for video and audio streaming based on Flash. Read5 is used for recording streaming audio data from microphone. The reason that a media server is needed is that recording sound from a web page directly is essential for Query-by-Clapping. It is a hard problem in web development to record audio and video, which has not yet been supported by the HTML5 standard.

Figure 4.1 illustrates the architecture of TMSE. The system is divided into two layers: the query layer and the database layer.

Users interact with the query layer, using different query inputs to express their intention towards tempo. The query layer receives and processes different user queries, and presents the results. All the other five query inputs are processed with corresponding algorithms,in the end only a tempo value is passed to the database layer. That is to say, Query-by-Sliding, Query-by-Tapping, Query-by-Example, Query-by-Clapping and Query-by-Walking are first transferred into

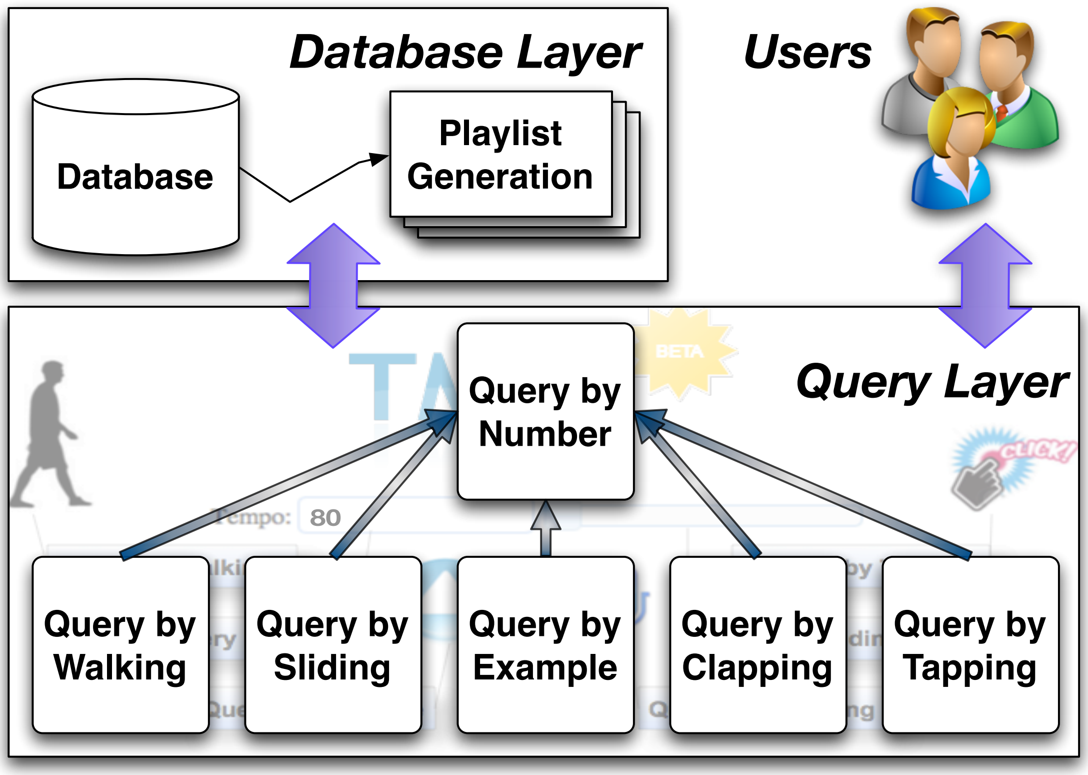


Figure 4.1: Architecture of TMSE

Query-by-Number in the query layer. Only Query-by-Number interacts between the query layer and the database layer.

The database layer is used to store the music information, together with their tempo values. The tempo values stored in the database are obtained from off-line processing. The best-performance beat tracking algorithm (i.e. Klapuri's) in our algorithm evaluation is used to process all the songs in our music database, and the median value of Inter-Beat-Intervals of each song is used to calculate the tempo value. We use MySQL as our database. A MySQL table containing the name of a song, the tempo value of a song, and the file path of a song.

4.3 Query-by-Number

Query-by-Number is designed to allow users to query songs with an exact tempo value. For Query-by-Number, the tempo value is passed to the database layer and all songs with similar tempi are retrieved into a *VIEW* (a temporal table in database). A playlist is generated from this VIEW, and songs with less tempi difference are ranked higher in the returned result.

4.4 Query-by-Sliding

For normal users, the exact tempo value could mean nothing to them, instead *fast/slow* tempi were more important to them. A slide bar is a good indicator of fast/slow, and this is why Query- by-Sliding was implemented as an individual query input in TMSE.

For implementation, Query-by-Sliding read the tempo value on the slide bar, and the tempo value is passed to Query-by- Number. When users are sliding, they can see the playlist changing accordingly with different tempi values.

Adding auditory feedback to Query-by-Sliding could be a useful feature that remains as a future work.

4.5 Query-by-Tapping

Query-by-Tapping allows users retrieve songs according to the speed of users mouse clicking.

For Query-by-Tapping, we used JavaScript on the web page to capture the movement of users clicking. The mean value of click intervals is used to calculate a tempo value.

If a user stops clicking the mouse for more than 1.5 seconds which corresponds to the slowest tempo of 40 BPM in our database, TMSE will consider it as a boundary of two consecutive queries.

4.6 Query-by-Example

Query-by-Example returns songs in the similar tempi of users example songs.

Users can upload an audio file with any audio formats using Query-by-Example. Furthermore, video files with audio channels are also supported. After uploading the file, the server invokes a shell script to convert the audio format into WAV using an open-source tool FFmepg [FFm]. Then the WAV file is processed using Daviess algorithm [DP07] to get the tempo value for Query-by-Number.

4.7 Query-by-Walking

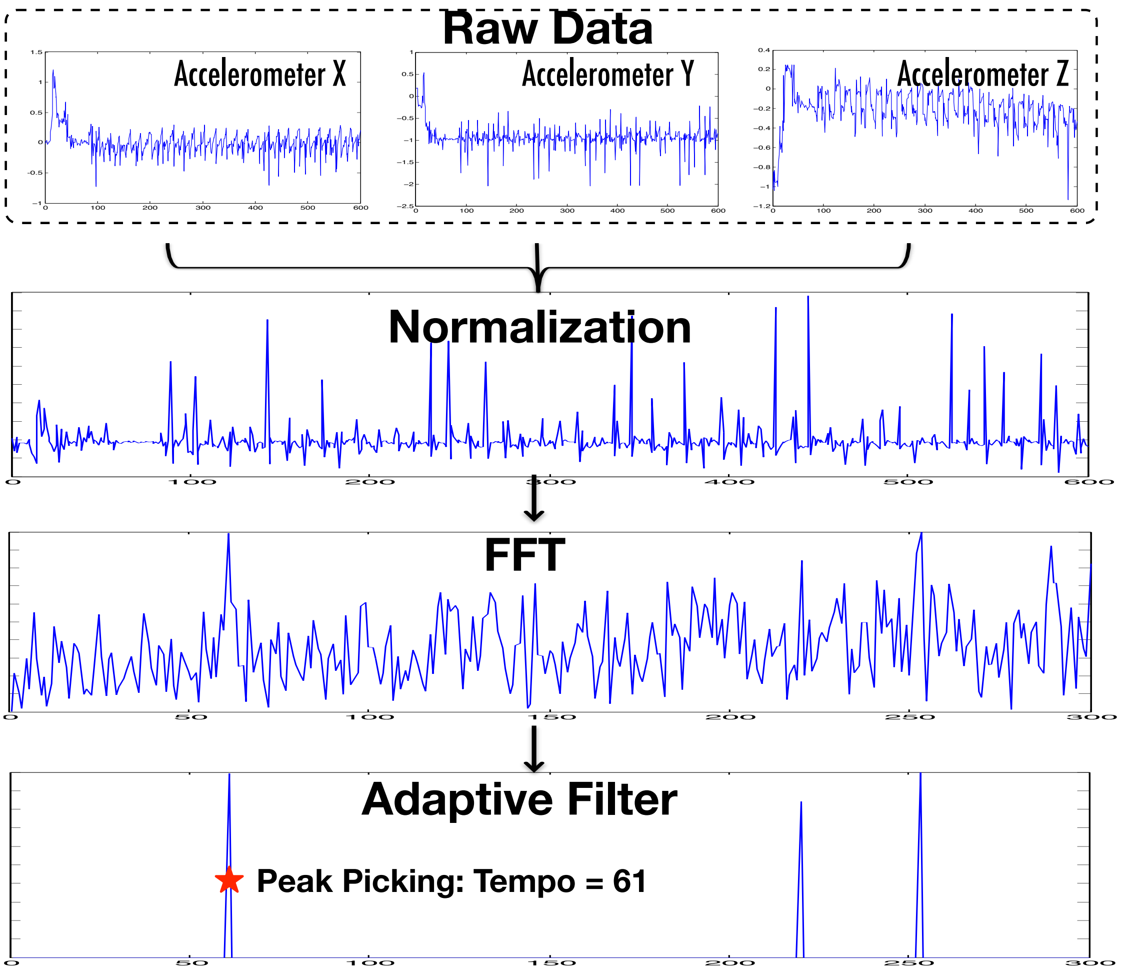


Figure 4.2: Tempo Estimation Based on Accelerometer Data

Query-by-Walking is designed for music therapists to search for suitable music for gait training program of a Parkinson's disease patient.

A patient can simply place an iPhone on his/her pockets and walk, and then the tempo of his/her walking could be estimated based on the accelerometer data of the iPhone. This tempo value will then be passed to Query-by-Number.

Figure 4.2 shows each step of our algorithm of estimating tempo value from accelerometer data. The accelerometer data from x, y, and z axes are normalized first using quadratic sum:

$$sensor(t) = x(t)^2 + y(t)^2 + z(t)^2$$

The normalized signal is transferred by a FFT routine. We analyze the signal in the frequency domain in order to find out the most dominant frequency, which corresponds to the frequency in which user walks.

The moving average is used as the threshold, and an adaptive filter is applied to the sensor signal. Only the most dominant frequency bins are picked as potential candidates.

Then a peak picking method is used to get the maximum of the candidate. The peak picking method make sure that the result frequency corresponds to a tempo value in the range of 0 - 200 BPM, for normally people walk less than 3 steps/second (180 BPM).

In the example shown in Figure 4.2, this algorithm outputs a tempo of 61, which is very close to human annotation.

The complete algorithm process is shown below:

```
% Load sensor data
load x.sensor;
load y.sensor;
load z.sensor;

% Normalization: sensor=x^2+y^2+z^2
sensor = x.*x + y.*y + z.*z;

% FFT & normalization
freq_sensor = abs(fft(sensor));
freq_sensor=f_sensor/max(f_sensor);

% Use moving average to do adaptive filtering
% The code below showed that it filters out
% values below o.5 and the window size is 10
peak = adaptiveThreshold(freq_sensor, 10, 0.5);

% Choose the peak which corresponding to 0~200BPM
tempo = tempoCal(peak);
```

4.8 Query-by-Clapping

For Query-by-Clapping, a user can simply clap his/her hands to search for music according to the tempo of his/her clapping.

Recording sound directly from the browser is essential for this query input, so we use Red5 as the media server. The Red5 media server stores the streaming data from microphone to an audio file.

After initial evaluation, we were surprised that all 4 beat tracking algorithms we employed were not reliable to estimate tempo of clapping for some reasons. For example, algorithms might output 120 BPM for clapping in 60 BPM. As a result, we have designed a simple peak-picking algorithm on time-domain.

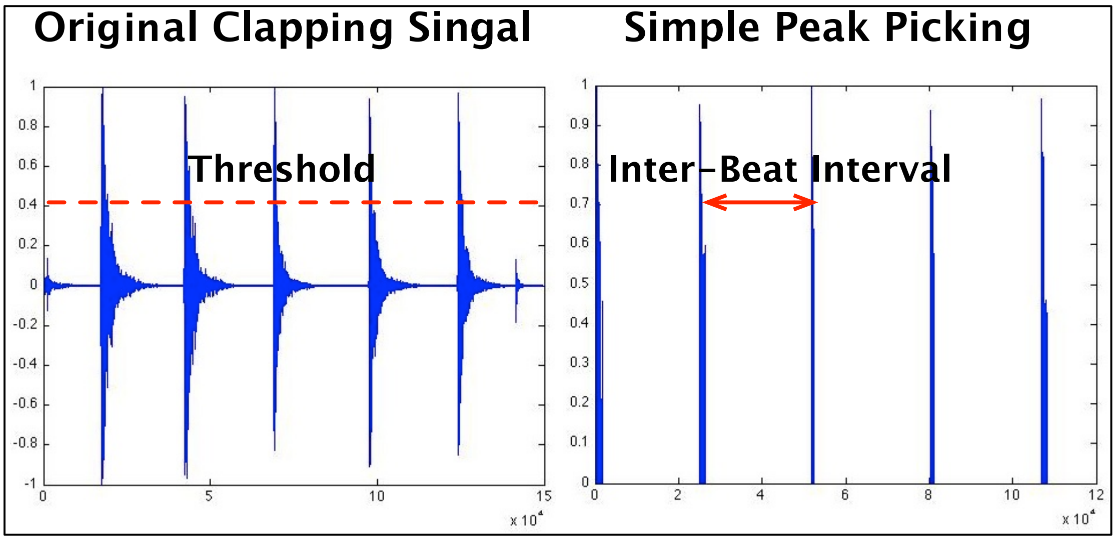


Figure 4.3: Clapping Signal Processing

Figure 4.3 illustrates the thresholding step of our clapping detection algorithm. After normalization, all amplitudes below 40% of the maximum amplitude are set to zeros. Then peak-picking is used to get beats and inter-beat intervals. The clapping tempo is derived from inter-beat intervals. The result is quite reliable in our preliminary study.

The complete algorithm process can be seen below:

```
% Loading wav file
rawHumming = wavread('clapping.wav');

% Normalizing the wav file
maxRawHumming = max(abs(rawHumming));
rawHumming = rawHumming/maxRawHumming;


% Thresholding
```

```
for i= 1 : size(rawHumming)
  if (rawHumming(i) < 0.4*maxRawHumming)
    rawHumming(i) = 0;
  end
end

% Cut the white part in head and tail of the signal
for i = 1 : size(rawHumming)
  if (rawHumming(i) ~= 0)
    wavbegin = i;
    break;
  end

end

for i = size(rawHumming) : -1 : 1
  if (rawHumming(i) ~= 0)
    wavend = i;
    break;
  end
end

rawHumming = rawHumming(wavbegin:wavend);

% Grouping each beat.
vector = [];
counter = 0;
last = 0;

for i = 2 : size(rawHumming)
  if (rawHumming(i)~=0)
    if (i - last > 1000)
      vector = [vector i];
      counter = counter +1;
    end
    last = i;
  end
```

```
end

vector = vector/44100;

% Using Interval median to calculate tempo
tempInterval = eInterval(vector);
tempIntervalMedian = median(tempInterval);
tempo = 60/tempIntervalMedian
```

4.9 Tempo Adjustment

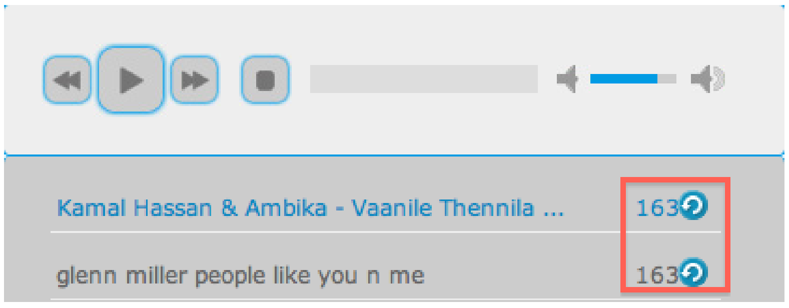

Kamal Hassan & Ambika - Vaanile Thennila ...
163
glenn miller people like you n me
163

Figure 4.4: Tempo Adjustments

To enhance the user experience, we have also implemented a tempo stretching algorithm [Hoc] without changing the pitch. As shown in Figure 4.4, the user can modify the music tempo from 163 to the desired tempo of 170 by clicking the icons beside the tempo.

Chapter 5

Evaluation

Our evaluation consists of 2 parts:

1. An evaluation on the accuracy of 4 beat tracking algorithms.

2. A preliminary user study on TMSE.

5.1 Evaluation of Beat Tracking Algorithms

For evaluation of beat tracking algorithms, we collected 787 songs from YouTube, and most of them are old songs because of the potential clinic usage of the system. Two amateur musicians were hired to annotate the whole music collections. Only 560 songs were used as the ground truth after validation and cross-checking.

We compared 4 different beat tracking algorithms on the ground truth. We used each algorithm to get all the inter-beat intervals on each song. We derived tempo values from the mean of the intervals. Results were compared to those in the ground truth.

Table 5.1: Accuracy of tempo estimation algorithms

	Davies [DP07]	**Dixon** [Dix07]	**Ellis** [Ell07]	**Klapuri** [KEA06]
AC1	0.651	0.492	0.140	**0.689**
AC2	0.823	0.824	0.646	**0.880**

Table 5.1 shows the result of the algorithms evaluation. AC1 and AC2 are different accuracy measurements. Let us assume that a is the tempo value of a song obtained from the algorithms, and b is the corresponding tempo value in ground truth. In AC1, only $a = b$ is acceptable. In AC2, three cases are acceptable: i) $a = b$, ii) $a = 2b$, and iii) $2a = b$.

Figure 5.1 is a detailed comparison of 4 algorithms. There are 3 bars for each algorithms: i) number of songs matched exactly to the ground truth; ii) number of songs matched to 2 or 1/2 times to the ground truth; iii) others, which we could consider as errors. As can be seen, Klapuri's algorithm performs the best: most of songs are exactly matched to ground truth, with some 2 or 1/2 times matched and a few errors. Davies' performs the second best. Ellis' performs the worst, most of its results are 2 or 1/2 times matched to ground truth, with most errors.

As shown in Table 5.1 and Figure 5.1, Klapuris algorithm achieved the best performance among all 4 algorithms. Therefore we chose it for the offline processing. For online processing (Query-by-Example), we use Davis algorithm, because it offers a better trade-off between accuracy and time efficiency than Klapuris algorithm.

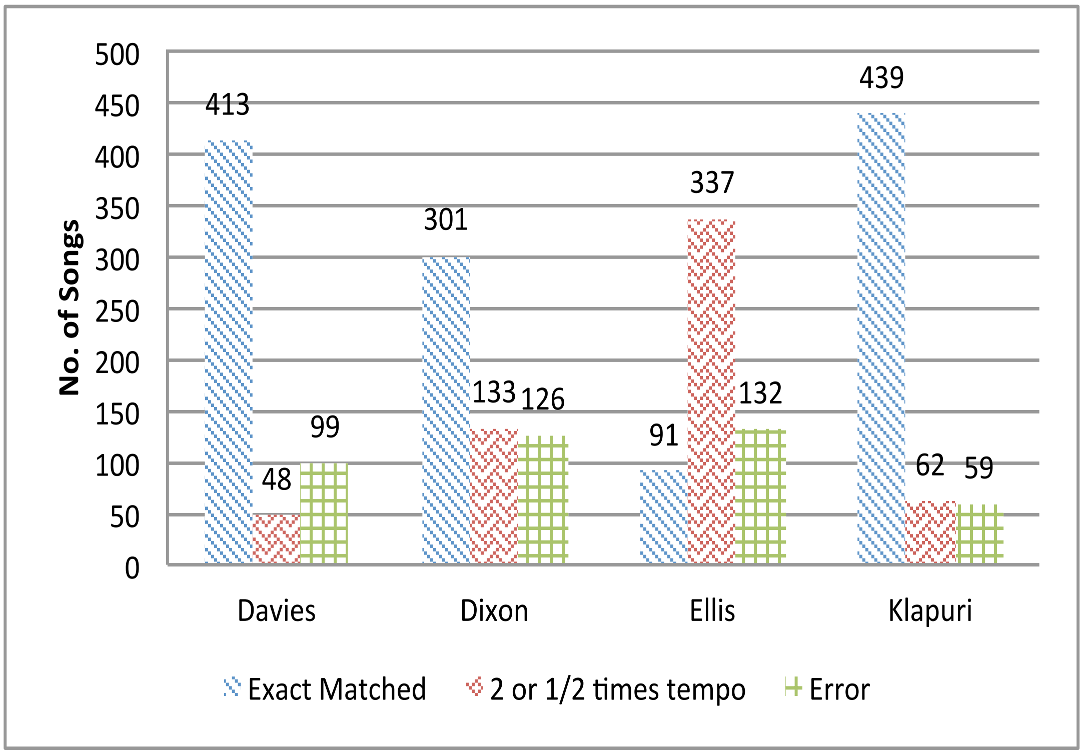


Figure 5.1: Detailed Comparison of 4 Algorithms

5.2 Preliminary User Study

We conducted a preliminary user study to validate our system, which focused on the tempo searchability of our TMSE system.

5.2.1 Evaluation Setup

Subjects

15 graduate students volunteered to participate. They are all fond of music. 9 out of 15 subjects use music search engines (e.g., YouTube) everyday, while the rest use them rarely. We conducted the study in a casual environment, so that users can feel relaxed and give genuine feedback.

Per-component evaluation

Because Query-by-Walking was designed particularly for potential clinic usage of gait training, which is an on-going project. Therefore, we just conducted a preliminary user study using Query-by-Number, Query-by-Tapping, Query-by-Sliding, Query-by-Example and Query-by-Clapping.

We arrange subjects to take a per-component evaluation. In the evaluation, subjects can try out different query inputs of TMSE, to search a song in tempo matched to their intention.

We asked users to listen to one sample song: Easy Love by Chinese singer Jay Chou. They were told to feel and remember the tempo of song while listening. Assuming that their intention is to search for songs with similar tempi as the query, they were asked to use each query input in a random order. In the process of evaluation, subjects can listen to the sample song anytime to remember the intention. After listening to the results returned by search engine, they were asked to rate their satisfaction with each input mode.

Questionnaire

Subjects are asked to complete a questionnaire regarding the tempo searchability of TMSE.

All subjects were asked to fill in their basic information such as age, gender, and music search engine experience in a questionnaire. Throughout the whole study, subjects were asked to rate each component and the tempo searchability based on easy they can express their intention towards their intention tempo.

5.2.2 Result and Analysis

We summarized our evaluation results of the per-component evaluation in Figure 5.2.

In the Figure 5.2, the highest to lowest ranking for all input modes are Query-by-Example, Query-by-Tapping, Query-by-Clapping, Query-by-Sliding, and Query-by-Number. It is interesting to know that the users satisfaction with the query input is proportional to the easiness that they can express the tempo. If they want to express the tempo of a specific song, they would prefer Query-by-Example and let the system find the most similar songs. Therefore, it is also not surprising to know that Query-by-Tapping is the second best input mode compared to Query-

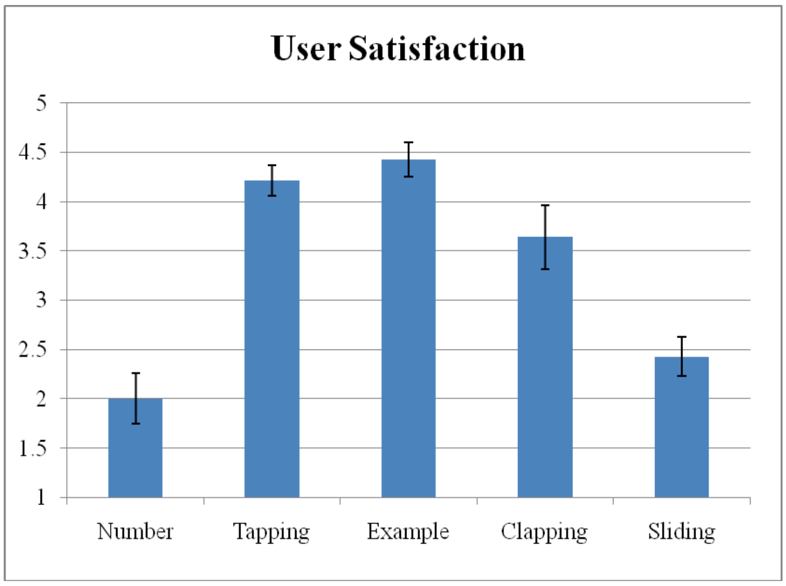


Figure 5.2: Per-component user satisfaction evaluation

by-Example, because users can easily tap along with the intended song. Query-by-Clapping is the third best, since clapping users' hands requires more sensory motor efforts than Query-by-Tapping. Query-by-sliding and Query-by-Number are the lowest two, A possible reason that users were unsatisfied with both two modes is that it was hard for them to express the tempo in the form of rigid number. Although Query-by-Sliding is better than Query-by-Number, which means a sense of fast/slow does help in users' tempo query formalization.

In the questionnaire, all subjects are content with TMSE about the tempo searchability. They indicated that the multimodal inputs of TMSE allows them form the query more easily than the search box only. In the questionnaire, many subjects also indicated TMSE can be a supplementary part to the existing music search engine.

The evaluation results indicate that in order to search for music with a specific tempo, TMSE provides a very useful and effective way, and the multimodal inputs can reduce users' intention gap.

Chapter 6

Future Work

In this chapter, we will explore some of the potential research topics in the future, based on what we have accomplished in this thesis.

Future work in this chapter will include two independent parts:

- **Future Work For iTap**: We present a detailed research plan for transferring iTap into GWAP.
- **Future Work For TMSE**: Three ideas based on TMSE are proposed, highlighting the motivations and the key research contributions.

6.1 Future Work For iTap

We intend to extend iTap as GWAP (Games-With-A-Purpose) in order to collect large amount of annotations. Here is a detailed research plan for this future work. A literature survey of GWAP is stated in Section 2.4

6.1.1 Motivation

We would like to design a mobile GWAP framework in an eyes-free environment. In this potential framework, we use mobile devices to collect user tapping in an eyes-free environment, and we design GWAP approaches to motivate users to annotate as many music pieces as possible. Users play this GWAP to have fun, and tapping generated serves as the human annotation as a side effect of the game play.

Our novelties and research contributions of this future research include:

1. In the HCI (Human Computer Interaction) aspect, designing mobile eyes-free GWAP problem is challenging and interesting, which could solve music tempo annotation problem.
2. Analysis of large amount of annotation data from real users, both online (for game purpose) and offline (for annotation purpose). This process can discover new mathematical models to define the accuracy of tapping data from users and to derive the final tempo values.
3. An evaluation on mobile eyes-free tempo annotation versus traditional PC-

based tempo annotation.

6.1.2 Research Plan

Some GWAP applications have been reviewed in section 2.4. The research plan for extending iTap includes the following steps:

1. Evaluation the reliability of eyes-free annotation;
2. Eyes-free mobile GWAP design;
3. Annotation Data Analysis.

Reliability of Eyes-Free Annotation

We intend to invent a new mechanism for eyes-free mobile tempo annotation. As the first step we need to prove that mobile eyes-free annotation is at least as reliable as PC-based methods.

In order to prove this, we will develop two prototypes for tempo annotation purpose on mobile and on PC respectively. In both prototypes, all the functions we need include: *Play/Stop* music, *Replay* for users to have another try if he couldn't follow the tempo at first, and *Skip* when the user cannot follow the tempo finally.

Then an evaluation needs to be done on the annotation reliability. We assign some number of songs (e.g. 10 songs) to some users, and they have to annotate all these songs both on mobile and PC. When they are annotating songs on PC, they do it in the traditional way, sitting there, listening to the music, and clicking their mice. When they are annotating songs on mobile, they are asked to walk around while listening to the music, and use as little visual attention as possible while tapping on the touch screens.

After annotating both on mobile and PC, if the statistic shows there is little different between two sets of results, it is more confident for us to say that annotation in a mobile and eyes-free environment is as reliable as in desktop environment.

Eyes-free Mobile GWAP Design

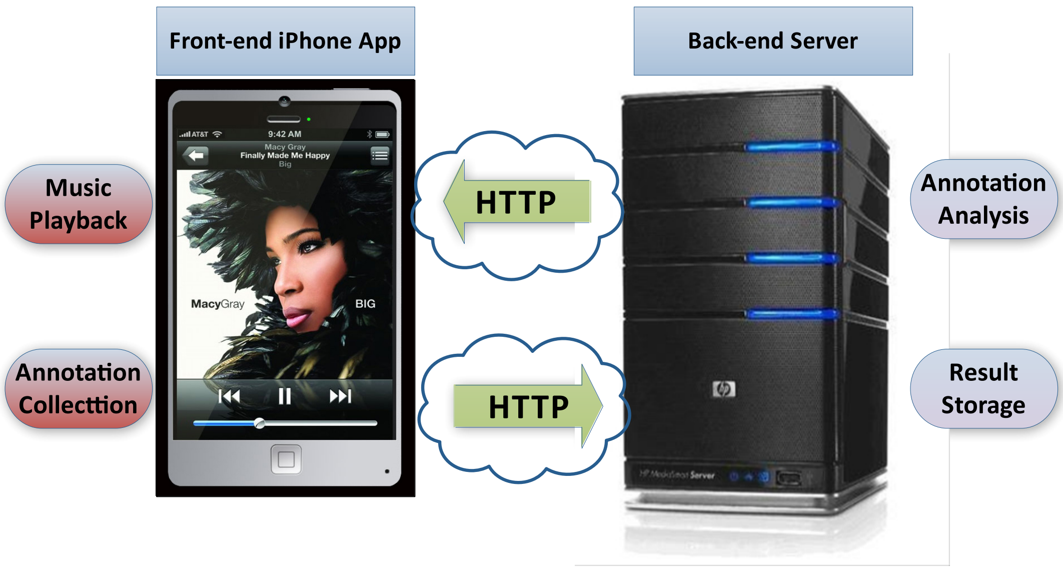


Figure 6.1: The architecture of GWAP

In our GWAP system, the front-end is iPhone or other mobile phones with touch screens are used to collect annotations, and the back-end is our server, which is used to store and analyze annotations. The system architecture is showed in Figure 6.1.

When users start playing our GWAP, audio data will be streamed from the server to users' phones. The front-end phones collect users' tapping while playing the music, and these tapping data as annotations will be sent back to the server.

Some research problems can be investigated here:

1. *Audio Streaming Efficiency*: How to stream audio data to many mobile phones efficiently?
2. *Low-power on Mobile*: How to save power consuming of phones in the game process?
3. *Real-time Parallel Data Analysis*: We are developing a game, and users always need real-time feedback. If there are many users playing the game at the same time, is that possible that we use some parallel framework (e.g. *MapReduce*) to do the real-time parallel data analysis?

However, the most challenging research problem is still designing an eyes-free GWAP. In section 2.4, most GWAP requires two or more users playing games together, for the purpose of evaluating their results. And all of them need users' visual attention all the time. In our design, it is very hard for two users annotating together, because listening and tapping is a *personal* natural response. Without visual attention, we cannot use most of the design elements from the traditional games.

However, we can still try some strategies for the eyes-free GWAP design:

1. *Advertisement*: We can tell users this game is used for "training your sense of tempo".

2. *Award Mechanisms*: A simple but useful way to motivate user is provided awards for them. For example, users can earn credits if they play the game, and users are in different levels according to their credits. Users in higher levels can get higher privileges in the game.

3. *Later Results Evaluation*: It is hard to let two persons play together, so that we can separate the process into the *participation step* and *evaluation step*. In participation step, user can earn some small credits (e.g. 100 points) when they are participating to annotate a song. And later in the evaluation step, users can get much more credits (e.g. 1000 points) when his annotation matched with former *or* later users. Furthermore, when a user gets in higher levels, which means his annotation could be more reliable, he can get more credits from every time of annotation.

4. *Special Functions*: When a user plays our GWAP more, more cool service he can receive from the game. For example, if a user can reach some certain level, then he can enjoy some hours of online radio based on tempo. When the user is working he can choose to listen smooth music with slow tempo, and when the user is running he can receive music with fast tempo. He can enjoy this kind of service for longer time by annotating more songs.

These are some aspects we have considered, and more mechanisms to motivate users will be figured out. Based on the whole design process, we could summarize some design principles in HCI aspect, about how to design eyes-free GWAP. These principles can be useful for other researchers.

Annotation Data Analysis

Annotation data analysis of music tempo could be an interesting and challenging problem.

First, we have many users' tapping data on each song, so we can try to build a statistic model of people's tapping. This could be a quantitative measurement of a one's sense of tempo. With this statistic model, we can tell how people act differently on a same song, and we can also judge the quality of one's annotation. Discovering on how one's sense of tempo changes could be another interesting problem.

Second, we have a lot of annotations now, and then we need to filter and purify them. The final goal is to build a well-annotated dataset, so discovering the "true tempo" from many people's tapping may not be a trivial job. We can use proper data mining methods to separate different sets of annotation, and derive the final most accurate tempo value for each song.

Third, for the game purpose, we need to do data analysis online in order to give users real-time feedback. So we can use time series analysis methods to deal with similar time series.

6.2 Future Work For TMSE

The main motivation behind TMSE is to reduce the *intention gap* for music search. Future work of TMSE can be in different directions:

6.2.1 Query-by-Walking

We intend to further develop Query-by-Walking to make it a useful clinic application for gait training of Parkinson's Disease patients.

Detecting the gait tempo (cadence) of patients would be the first step.

Nowadays more and more mobile phones come with multiple sensors. Instead of using accelerometer along, using multiple sensors to capture and detect gait could be an interesting and challenging research problem. More gait detection algorithms using different types of sensors will be experimented.

If we can detect the gait reliably, we can turn the research focus into *human-center design* and *personalization* for the patients:

Human-Center Design: Patients suffering Parkinson's Disease are incapable in steady walking. Could we design the user interface so they can use easily?

Personalization: Only accurate tempo is not enough to let patients enjoy the training process. Could we find the music which is not only in an accurate training tempo, but also enjoyable by the patients?

6.2.2 Auditory Query Suggestion

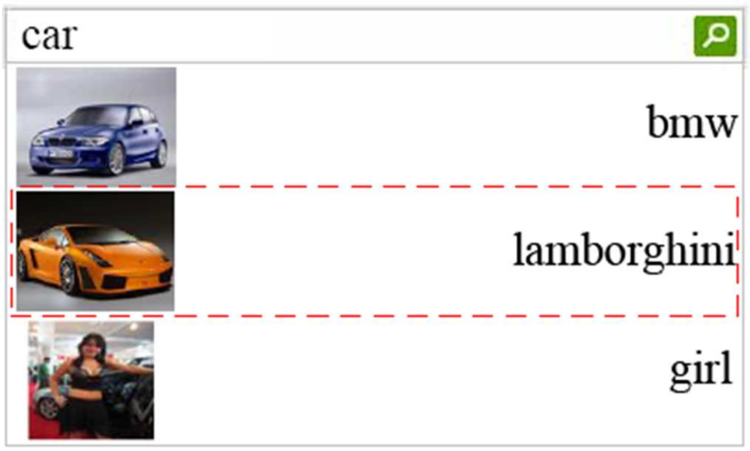


Figure 6.2: Visual Query Suggestion [ZYM+09]

The intention gap of music search also could be reduced when users input their queries. As can be seen in Figure 6.2, *Visual Query Suggestion* [ZYM+09] is a very successful approach to reduce intention gap when users are typing their queries for images. Similarly, we could try to develop *Auditory Query Suggestion*, which allows users to hear different example results, to reduce intention gap for music search.

One possible usage of the auditory feedback is in our Query-by-Sliding. Query-by-Sliding is designed to let users have a sense of fast/slow. However, right now we only provide the visual feedback. If an auditory feedback is provided, i.e. an metronome with fast/slow "bi-bi" sounds, then users will have a better understanding what does a tempo sounds like. This could be a reliable way to reduce users' intention gap.

6.2.3 Reducing Intention Gap in MIR

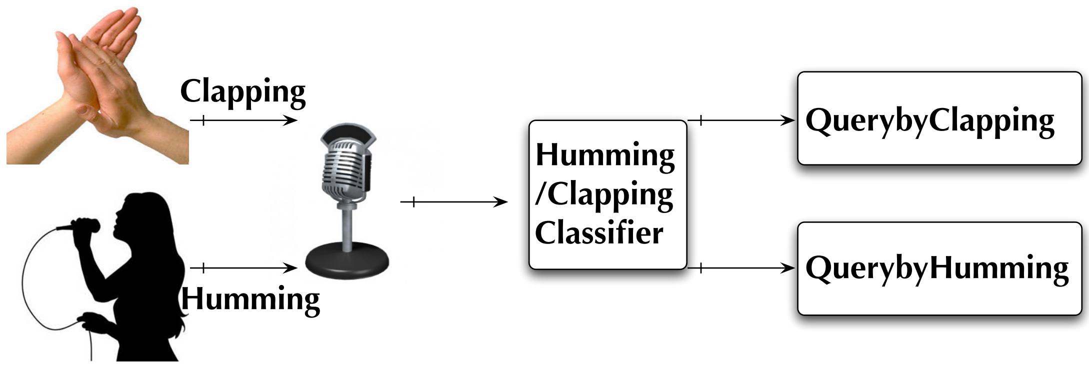


Figure 6.3: Sound-based Music Query

Figure 6.3 shows a possible project. No matter Query-by-Humming or Query-by-Clapping, sound-based music query is always popular: it is natural for people to retrieve sound through making sound to the search engine. We can merge Query-by-Humming and Query-by-Clapping into one interface, using a binary classifier. The classifier tells the search engine whether the input sound is clapping or humming. If the input is clapping, then the search engine will invoke Query-by-Clapping and return songs in similar tempi according to the speed of the clapping. If the input is humming, then the search engine will invoke Query-by-Humming and return songs in similar melody. This could further reduce the intention gap for users.

Chapter 7

Conclusion

We have presented TMSE, a tempo-sensitive music search engine with multimodal inputs, including Query-by-Number, Query-by-Sliding, Query-by-Tapping, Query-by-Example, Query-by-Clapping and Query-by-Walking, to reduce intention gap when a user formulates a tempo-based query.

We have designed an intuitive and simple-to-use user interface. To validate the proposed prototype, we carried out a preliminary user study consisting of a per-component evaluation and a questionnaire about searchability of TMSE. The results showed that Query-by-Tapping and Query-by-Example were the most satisfying and efficient in searching music based on tempo.

We have evaluated 4 beat tracking algorithms, and have selected the top 2 best performing algorithms for our system implementation.

We have also developed iTap, an eyes-free tempo annotation tool, to annotate our music collections.

In our future work, which states in Section 6.1 and Section 6.2, we intend to extend iTap as GWAP application to make the tempo annotation more easily, and we also intend to use TMSE in clinic application for music therapists.

Bibliography

[AD90] P.E. Allen and R.B. Dannenberg. Tracking musical beats in real time. In *Proceedings of the 1990 International Computer Music Conference*, pages 140–143. Citeseer, 1990.

[BEC+07] N. Bach, M. Eck, P. Charoenpornsawat, T. Köhler, S. Stüker, T.L. Nguyen, R. Hsiao, A. Waibel, S. Vogel, T. Schultz, et al. The CMU TransTac 2007 eyes-free and hands-free two-way speech-to-speech translation system. In *Proc. of the International Workshop on Spoken Language Translation*. Citeseer, 2007.

[Bro05] W. Brodsky. The effects of metronomic pendular adjustment versus tap-tempo input on the stability and accuracy of tempo perception. *Cognitive processing*, 6(2):117–127, 2005.

[BWE93] S.A. Brewster, P.C. Wright, and A.D.N. Edwards. An evaluation of earcons for use in auditory human-computer interfaces. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 222–227. ACM, 1993.

[Dix01] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30:39–58, 2001.

[Dix06] S. Dixon. Onset detection revisited. In *Proceedings of the 9th International Conference on Digital Audio Effects*, pages 133–137, 2006.

[Dix07] S. Dixon. Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–50, 2007.

[DP07] M.E.P. Davies and M.D. Plumbley. Context-dependent beat tracking of musical audio. *Audio, Speech and Language Processing, IEEE Transactions on [see also Speech and Audio Processing, IEEE Transactions on]*, 15(3):1009–1020, 2007.

[DPB00] C. Drake, A. Penel, and E. Bigand. Tapping in time with mechanically and expressively performed music. *Music Perception*, pages 1–23, 2000.

[Ell07] Daniel P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[FFm] FFmpeg. `http://www.ffmepg.org`.

[GKD$^+$06] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(5):1832–1844, 2006.

[Got01] M. Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, 2001.

[GS91] W.W. Gaver and R.B. Smith. Auditory icons in large-scale collaborative environments. *ACM SIGCHI Bulletin*, 23(1):96, 1991.

[HC03] H. Harb and L. Chen. A query by example music retrieval algorithm. In *Digital media processing for multimedia interactive services: proceedings of the 4th European Workshop on Image Analysis for Multimedia Interactive Services: Queen Mary, University of London, 9-11 April 2003*, page 122. World Scientific Pub Co Inc, 2003.

[HH03] S.W. Hainsworth and S.W. Hainsworth. Techniques for the automated analysis of musical audio. 2003.

[Hoc] J. Hockman. `http://www.kichiki.com/WAON/pv.html`.

[HR09] P. Hanna and M. Robine. Query by tapping system based on alignment algorithm. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 1881–1884. IEEE, 2009.

[HVA09] S. Hacker and L. Von Ahn. Matchin: eliciting user preferences with an online game. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1207–1216. ACM, 2009.

[JLY01] J.S. Jang, H.R. Lee, and C.H. Yeh. Query by tapping: A new paradigm for content-based music retrieval from acoustic input. *Advances in Multimedia Information ProcessingPCM 2001*, pages 590–597, 2001.

[KEA06] A. P. Klapuri, A. J. Eronen, and J. T. Astola. Analysis of the meter of acoustic musical signals. *Audio, Speech and Language Processing, IEEE Transactions on [see also Speech and Audio Processing, IEEE Transactions on]*, 14(1):342–355, 2006.

[KL02] H.M. Kamel and J.A. Landay. Sketching images eyes-free: a grid-based dynamic drawing tool for the blind. In *Proceedings of the fifth international ACM conference on Assistive technologies*, pages 33–40. ACM, 2002.

[Lar95] E.W. Large. Beat tracking with a nonlinear oscillator. In *Working Notes of the IJCAI-95 Workshop on Artificial Intelligence and Music*, volume 24031, 1995.

[LBH08] K.A. Li, P. Baudisch, and K. Hinckley. Blindsight: eyes-free access to mobile phones. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1389–1398. ACM, 2008.

[LVADC03] E.L.M. Law, L. Von Ahn, R.B. Dannenberg, and M. Crawford. Tagatune: A game for music and sound annotation. In *International Conference on Music Information Retrieval (ISMIR07)*, pages 361–364. Citeseer, 2003.

[LXH$^+$10] Z. Li, Q. Xiang, J. Hockman, J. Yang, Y. Yi, I. Fujinaga, and Y. Wang. A music search engine for therapeutic gait training. In *Proceedings of the international conference on Multimedia*, pages 627–630. ACM, 2010.

[LYZ01] L. Lu, H. You, and H.J. Zhang. A new approach to query by humming in music retrieval. In *Proceedings of the IEEE International Conference on Multimedia and Expo*. Citeseer, 2001.

[OFM06] N. Oliver and F. Flores-Mangas. Mptrain: a mobile, music and physiology-based personal trainer. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 21–28. ACM, 2006.

[Pau02] S. Pauws. Cubyhum: A fully operational query by humming system. In *Proceedings of ISMIR*, pages 187–196. Citeseer, 2002.

[Red] Red5. `http://www.red5.org/`.

[Ros92] D.F. Rosenthal. Machine rhythm–computer emulation of human rhythm perception. 1992.

[Sch98] E.D. Scheirer. Tempo and beat analysis of acoustic musical signals. *The Journal of the Acoustical Society of America*, 103:588, 1998.

[TEC02] G. Tzanetakis, A. Ermolinskyi, and P. Cook. Beyond the query-by-example paradigm: New query interfaces for music information retrieval. In *Proceedings of the 2002 International Computer Music Conference*, pages 177–183. Citeseer, 2002.

[TMR$^+$96] M. H. Thaut, G. C. Mcintosh, R. R. Rice, R. A. Miller, J. Rathbun, and J. M. Brault. Rhythmic auditory stimulation in gait training for parkinson's disease patients. *Movement Disorders*, 11(2):193–200, March 1996.

[TWV05] R. Typke, F. Wiering, and R.C. Veltkamp. A survey of music information retrieval systems. In *Proceedings of the 6th International Conference on Music Information Retrieval*, pages 153–160. Citeseer, 2005.

[TYW05] W.H. Tsai, H.M. Yu, and H.M. Wang. A query-by-example technique for retrieving cover versions of popular songs with similar melodies. In *Int. Symp. on Music Information Retrieval (ISMIR)*, pages 183–190. Citeseer, 2005.

[VA06] L. Von Ahn. Games with a purpose. *Computer*, 39(6):92–94, 2006.

[VAD04] L. Von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.

[VAD08] L. Von Ahn and L. Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.

[vAGKB07] L. von Ahn, S. Ginosar, M. Kedia, and M. Blum. Improving image search with phetch. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4. IEEE, 2007.

[VAKB06] L. Von Ahn, M. Kedia, and M. Blum. Verbosity: a game for collecting common-sense facts. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 75–78. ACM, 2006.

[VALB06] L. Von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64. ACM, 2006.

[Wan06] Avery Wang. The Shazam music recognition service. *Commun. ACM*, 49(8):44–48, August 2006.

[wAOSWFiP] web.py: A Open-Source Web Framework in Python. `http://webpy.org`.

[Web] Midomi: A Social Music Search By Humming Website. `http://www.midomi.com`.

[YZW11] Y. Yi, Y. Zhou, and Y. Wang. A tempo-sensitive music search engine with multimodal inputs. In *Proceedings of the international conference on Multimedia Workshop on MIRUM*. ACM, 2011.

[ZDC$^+$07] S. Zhao, P. Dragicevic, M. Chignell, R. Balakrishnan, and P. Baudisch. Earpod: eyes-free menu selection using touch input and reactive audio feedback. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1395–1404. ACM, 2007.

[ZYM$^+$09] Z.J. Zha, L. Yang, T. Mei, M. Wang, and Z. Wang. Visual query suggestion. In *Proceedings of the seventeen ACM international conference on Multimedia*, pages 15–24. ACM, 2009.