AutoDJ: An Adaptive Music Player Utilizing Computer Vison and Machine Learning

Michael Rosendahl Schmidt

2013

Contents

Chapter 1

Introduction

We should consider every day lost on which we have not danced at least once.

- Friedrich Nietzsche

The closely linked subjects of music and dance have been an integral part of human life for a long while and all along technology has continuously been improving the access to music. The possibility of recording sound with the first phonographs opened up for listening to music without being in a room with a musician and allowed for mass distribution. Radio broadcasting further increased the distribution possibilities by allowing users to listen to music without having to acquire the recording themselves. With the digital options of computers and MP3 players one can store thousands of music tracks in very little physical space and access it instantly. Through the recent advances in streaming media, access to millions of music tracks is now possible and these services can even be accessed from anywhere via one's smartphone. All this means that distribution of music and access to it are to a large extent challenges which are well on their way to being solved.

With the massive amount of music available, a challenge which is as relevant as ever is the choice of which music to listen to. Historically the set of tracks to play was pre-defined, the user only chose which recording to put on, which radio channel to listen to, or which concert to attend. Recent advances have given us systems which provide recommendations by looking at which music one has listened to in the past, which music one's friends are listening to, or by letting the user input a set of preferences. While these systems go a long way towards solving the challenge, most of them still lack the ability to react to the changing moods of a user or any kind of dynamic in the surroundings of where the music is played.

At a party venue the music is an important part of the experience and therefore the right music needs to be played. Figuring out what 'right music' means is a particularly hard problem as it is matter of sensing the atmosphere and finding an answer which fits multiple people. All this can be taken care of by a professional Disc-Jockey(DJ), as the job of a DJ is to make it a fun party and to get people to dance by playing the right music. Hiring a DJ is often not a option due to financial or practical reasons and one therefore has to rely on more manual methods of controlling the music.

This report documents the development of AutoDJ, an intelligent system with the vision of

automating the selection of music to be played at a venue and thereby make redundant the need of a regular DJ. This should be possible by using sensors to monitor the party and then applying machine learning to that set of data. Replacing a human DJ is a massive undertaking but for a start a more intermediate goal would be to improve on the experience compared to a pre-defined playlist. Even with more humble goals, a foundation of knowledge is needed which means research on related areas to uncover the problem space and to learn from state of the art projects. This report document the learnings and the choices that had to be made along the way in the attempt to build the AutoDJ system.

The report is structured as follows: First an analysis on the problem space is carried out touching upon areas such as music consumption, DJ'ing, ambient intelligence, playlists, and adaptive systems. Existing attempts at playlist generators are evaluated to learn from state of the art systems. With a foundation of knowledge on the related areas, the report continues with a chapter on the design considerations for the AutoDJ system. It is then described how this system is implemented and tested with users at a party venue. The report ends with a discussion of the results and a lookout to further improvements.

Chapter 2

Previous Work

2.1 Music Consumption

Seeing how this project deals with music it is relevant to start out with a look at how music is typically consumed. Music listening can either by an entirely manual operation by picking each music track or it can be predefined such as with radio and concerts. Jukeboxes are one of the first examples of a direct interface for letting users pick the music in public venues[Gol03]. The first editions appeared around the end of the 19th century, although the term was first around 1940, and could play a user picked track on coin insertion. Some editions introduced popularity counters which allowed the owners to keep the popular records and replace the lesser-played songs.

When it comes to predefined sets of music, the most important phenomena is the playlist, a sequence of music tracks with a specific playback order[FL10]. A playlist can take form as an album, mix tape, radioshow, and other forms depending on which media it is used in. Although a regular CD album may be considered a playlist, the term usually denotes a combination of music from different artists. This idea with a curated a set of music tracks began around 1850 in London, here with live performances of multiple composers [Web01]. With the later appearance of radio this was continued with playlists based both around genres and hand-picked by radio hosts [BB00].

For music playback on digital devices, popular music players on the PC are programs such as Winamp[1] and iTunes[2] which allow a user to listen to music from his private collection. Streaming is becoming very popular due to the instant access to millions of tracks without having to worry about buying them or having the disk space. Popular examples include Pandora[3], Spotify[4], Grooveshark[5] and Last.FM[6], most of which are also able to provide personalized recommendations. Unlike Last.FM and Pandora, Spotify is built around a downloadable client. A key feature of Spotify is that one can share and collaborate on playlists.

[1] `http:/winamp.com`
[2] `http://apple.com/itunes`
[3] `http://pandora.com`
[4] `http://spotify.com`
[5] `http://grooveshark.com/`
[6] `http://last.fm`

An overview of music playback systems can be seen in figure 2.1. As seen, music players have taken on many new areas to improve upon the old manual and non-social selection of music. Today systems can learn from one's friends and make automated recommendations based on one's preferences and advanced music classification systems.

Figure 2.1: Music player systems and their focus[FL10].

2.2 Playlists

Before going into more complex systems, an area to explore further is playlists as those are at the core of curated music experiences. While the concept of a playlist is simple, there are many rules and approaches when it comes to shaping one.

> *To me, making a tape is like writing a letter — there's a lot of erasing and rethinking and starting again. A good compilation tape, like breaking up, is hard to do. You've got to kick off with a corker, to hold the attention (I started with "Got to Get You Off My Mind", but then realized that she might not get any further than track one, side one if I delivered what she wanted straightaway, so I buried it in the middle of side two), and then you've got to up it a notch, or cool it a notch, and you can't have white music and black music together, unless the white music sounds like black music, and you can't have two tracks by the same artist side by side, unless you've done the whole thing in pairs and...oh, there are loads of rules.*
>
> - Nick Hornby, in High Fidelty [Hor96]

Creating a playlist of music is commonly done either by manual selection or via shuffle (random filling). While the latter at first appears as a completely naive approach, it has gained much popularity due to reasons such as "had me re-examine things I thought I knew about my favorite music" [LVH05]. Shuffle can also be used to invoke certain affective responses such as feeling refreshed [LVH06]. A study shows that most users with a digital music library

prefers to listen to it using shuffle, as seen in figure 2.2.

		content organisation		
		constrained	unconstrained	
preferred listening	shuffle	22	69	91
	both	4	4	8
	sequential	13	1	14
		39	74	113

Figure 2.2: Preferred listening mode. Constrained refers to some kind of structured sub-section of their music library [LVH06].

Many different factors such as variety, coherence, coolness, and context are among the factors to consider when creating a good playlist. The importance of each has been evaluated in an user test by [DMV97] with 14 parcipants. The participants were first given an hour to create a playlist and after after that they filled out a questionnaire. The importance of each factor can be seen in figure 2.3 and should be considered as containing a certain amount of uncertainty due to the number of participants and any self-reporting error.

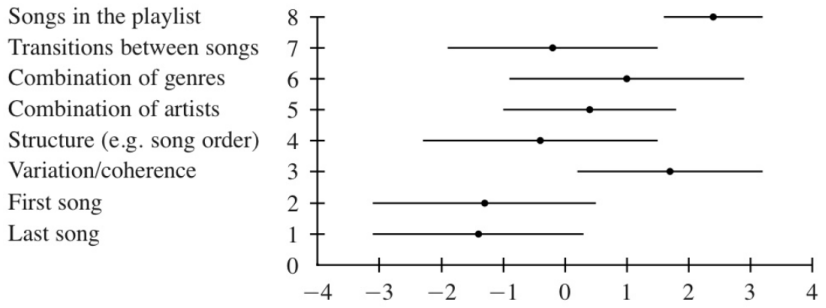

Figure 1: Importance of various factors in creating a playlist.

Figure 2.3: The importance of factors[DMV97]. The bars represent standard deviation.

Work has been done on social playlists and their implications on friendships. [LR08] investigates what happens if a group of friends are forced into listening to a mix of tracks from the music libraries of both themselves and their friends, their participants report that between 30-50 percent of the songs are "bad songs". This underlines the fact that sharing a playlist amongst multiple people does inevitably result in not being able to please everyone at all times.

One of the places where music is of high importance is in nightclubs and other party venues. Not much information exists on music is handled but presumably it is some mix of bar managed stereos, juke boxes and DJs. One example of how it can be handled is by using color coded mix tapes, each with a different kind of music[OLJ+04]:

> *Three colour-coded categories loosely designating when they are supposed to be played – "Green" for the daytime, "Yellow" for weekday evening, and "Red" for Friday and Saturday nights. "Green" music is subdued and relaxed background*

mood music and more "middle of the road". "Yellow" music is slightly more upbeat and "Red" is livelier still.

All this goes to show that there is lots of work being put into creating good playlists and there many considerations to make. The next section will evaluate how a DJ handles the task of deciding which music to play.

2.3 DJ'ing

Many nightclubs employ DJs to control the music selection and playback. One of his most important tasks is to continuously adapt the playlist to suite the audience and it is therefore relevant to look at the basis on which he performs this task.

The DJ, and other similar roles as curator of music, has had a central position in music and society from long before the first recordings were made[BB00]. The general history of dance shows that DJ-like persons have been appearing in all ancient cultures, although very different from culture to culture. It has been found that the job of a DJ involves many roles such as entertainer, record collector, intuitive psychologist, producer and musician. The claim of [BB00] is that the most important part of being a DJ is understanding moods and shaping moods. It should be noted that [BB00] does not deduct this claim based on any scientific method. Other sources report that says that DJ'ing is about finding an optimal ordering of tracks, the so called process of sequencing [Cli00]. From this paper it is also reported that a typical DJ session lasts between 40 minutes and 6 hours. Their claim is that a common goal for a DJ is to provide incentive to buy more drinks.

The area of what the work of a DJ is seems underrepresented in the current body of research literature and an interview was therefore held with Jacob 'ScratchMagic' Poulsen to get further details on the work of a DJ [Pou13]. He is a DJ with experience from 15 years of DJ'ing at the most popular venues in Copenhagen and as a member of the critically acclaimed band 'Nobody Beats The Beats'.

In the interview Jakob agrees with [BB00] that DJ'ing is very much about sensing moods and wanting to shape them. He operates mostly on with the objective of maximizing activity on the dance floor but sometimes also receives special requests to create a special kind of atmosphere which not necessarily involves dancing. When evaluating the reception of a played track he mostly looks at the amount of people dancing and how they dance. Some hints can also be picked up by looking at the glances from the crowd. This selection of tracks can be very much influenced by what a given crowd wants or more what suits the DJ in question, this differs a lot depending on the type of DJ.

Jacob typically starts out with playing music that is soft and slow in tempo and then works his way towards harder music that is faster in terms of BPM[7] On the topic of tempo, [Cli06] recommends similar a approach to tempos as shown in the graphs in figure 2.4. Jakob also finds that people rarely want to dance early in the night and it is mostly just a question of waiting for them to get sufficiently drunk. There are cases where he puts on one of a

[7]Beats Per Minute, a measurement of the tempo or speed of a piece of music. Typically slow music is below 80BPM, moderate speed between 80 and 100BPM, and fast music is above 100BPM.

few selected tracks which based on his experience knows to be effective at getting people on the dance floor. Still, it is a major issue to get the first people on the dance floor and it is something he tells that all DJs struggle with.

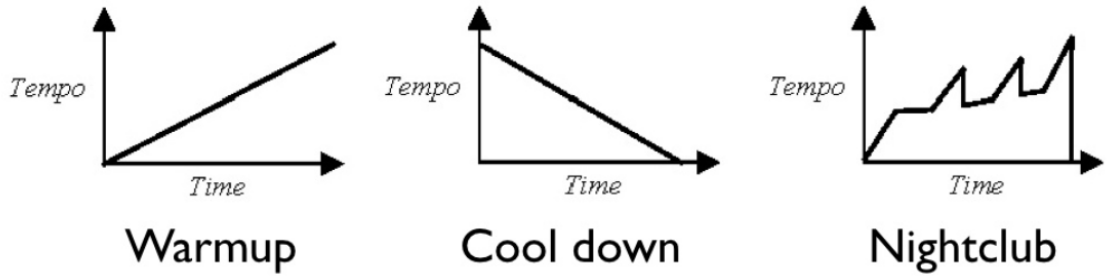

Figure 2.4: Tempo trajectories in three different situations. The left-hand graph shows a the trajectory for a warmup set of tracks to be played as the first set. The center graph shows a cool down set which can be used to signify the ending of a party. The right-hand graph spans over longer time and shows how tempos could evolve over the course of a night at a nightclub venue [Cli06].

2.4 Ambient Intelligence

The previous sections covered the basis of music consumption and various examples of both digital and human kind for playlist generation. As found in the previous section, an important part of DJing is to sense the atmosphere and mood of a given venue. It is therefore needed to investigate how a digital system can be developed with features similar to the eyes and ears of a DJ. For this purpose the area of ambient intelligence is relevant, as it deals with electronic environments that are sensitive and responsive to the presence of people. This is often also referred to as smart environments and ubiquitous computing. An important feature and challenge for interaction design in these environments is that the interaction is implicit and unintentional, although it works to the benefit of the user, when done right[HML09].

There are said to be five characteristics of ambient intelligence[Den01]:

- Embedded: many networked devices are integrated into the environment
- Context aware: these devices can recognize you and your situational context
- Personalized: they can be tailored to your needs
- Adaptive: they can change in response to you
- Anticipatory: they can anticipate your desires without conscious mediation.

The embedded sensors can measure many different types of input including temperature, pressure, audio, and many more. One of the more advanced sensors is computer vision which allows computers to handle visual input. An example of an ambience intelligence system with computer vision can be found in [CHR+00] which presents a modular system for real-time analysis of body movement and gesture. They show how this can be applied for control and

generation of sound, music, and actuators. What they are trying to do is to give an agent a similar view of the world compared to what a human has. One of the things they look at is basic dance theory and try to construct information processing which deals with evaluations based on taste and feeling. A clear challenge with their project is that it appears to require excessive amounts of expensive hardware in order to perform.

2.5 Adaptive Systems

Each party has a specific setting with a unique set of users and an ever-changing atmosphere which means that an adaptive system is required. As found in the previous chapter, adaptiveness is also a core characteristic of ambient intelligence systems. When it comes to adaptive systems one typically talks about systems featuring artificial intelligence or machine learning. Machine learning is a branch of artificial intelligence (AI) which deals with systems that learns from data through training[MCM86]. It focuses on prediction based on known properties and is closely related to data mining, which focuses on discovery of unknown knowledge.

One of the most important applications of machine learning is recommendation systems, these help users find items or social elements that match their specific needs [LSST]. Recommendation systems usually compare a user with some reference, and then try to predict the "rating" a user would give to an item they had not yet considered. In terms of this project, the item would be a music track and the rating would be the user's desire to be on the dancefloor while that track is playing. The references are usually from either the user's social environment (collaborative filtering (CF)) or the item itself (content-based) or [MLDLR03].

Collaborative filtering measures the similarity of a profile between agents and recommend what similar agents have already chosen. [SKKR00] performs an analysis on recommender systems for e-commerce and finds that CF-based algorithms are able to provide high-quality recommendations. It is also worth noting that several high-profile companies have CF-based algorithms heavily integrated into their business models, e.g. Amazon and Netflix. An example from music space is Last.FM which is designed around collaborative filtering with the users having "Love", "Skip", and "Ban" buttons to provide feedback.

A problem with CF is how to make recommendations when a user is new, ie. the system has no information to base its recommendations on, this is called the cold start problem. This can to be some extend be helped by having a sufficiently large knowledge base. It is commonplace today to already have information stored on one's music preferences, this is often the case on e.g. Spotify or Facebook. This can potentially be extracted from the crowd at a venue and used as a base for the music selection and thereby solving the cold start problem.

As mentioned, a different approach to these systems is with content-based recommendations. A content-based filtering system selects items based on the correlation between the content of the items and the user's preferences as opposed to a collaborative filtering system that chooses items based on the correlation between people with similar preferences. The user preferences can be found either explicitly by asking the user or by implicit learning, the latter which is possible with adaptive systems.

To be able to evaluate a music track for selection via a content-based approach, some properties of the track need to be known in order to compare them against the user preferences. The

properties, or attributes, of a music piece can be either found from low-level audio features that describe the audio signal (the waveform) itself or as more high level descriptions.

The Music Genome Project [GWS+06] was started with an attempt to classify songs using almost 400 attributes, ranging from very low-level to very high-level features. This work became the foundation for Pandora Radio, which at its core is an internet radio but with the unique feature being its personalized music discovery service, the first of its kind when launched. Another project which offers meta data for millions of tracks is Echonest, which is accessible via an online web service [BMEWL11]. One of the ways of defining these attributes is by 'tags', a high-level property which can be any kind of term that describes a piece of music. The tags on Last.FM are found to typically be related to genre, location, and mood [BMEM10]. These tags can also be created automatically, although the result is often not of the required accuracy.

2.6 Automatic Playlist Generation

With a base knowledge on the areas related to the song preference prediction, it is now possible to look closer at the systems dealing with prediction of what a user wants to listen to, these kind of systems are commonly named playlist generators. Such systems try to tackle the fact that through the recent rise in amount of music that many users have available, it has be become an increasingly unclear task to manually choose which music track that suits one's preferences in a given situation. This section will look at systems which can compose an immutable playlist based on an initial set of input.

A straight forward and popular commercial example of a playlist generator is the iTunes Smart Playlist feature, as seen in figure 2.6. It allows a user to specify a range of criteria such as rating, play count, time since last played, etc. Furthermore it can be set to be updated live whenever the user adds manipulates his music collection. Another more automatic system is found in Spotify which features recommendations of both playlists and tracks based on location, as seen in figure 2.5.

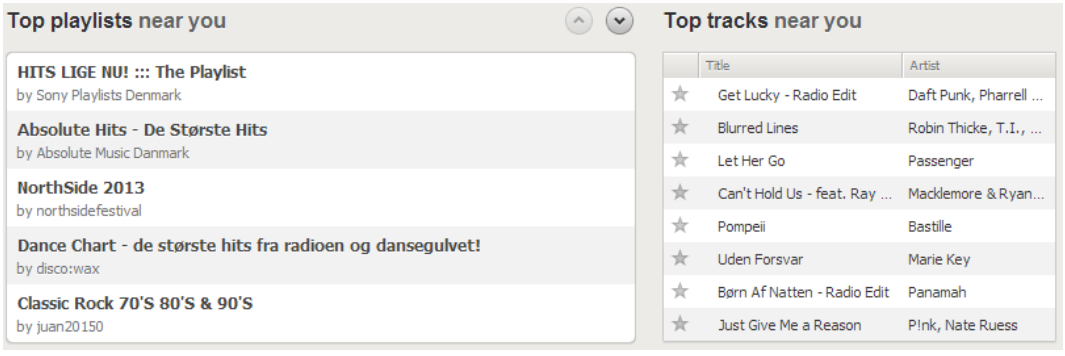

Figure 2.5: Spotify location based recommendations.

[MED+09] generate playlists based on data learned from a large set of data from professional radio station playlists. They use audio-based features as input to their model and use this to generate a similarity space of which songs that fit well together. A user can furthermore

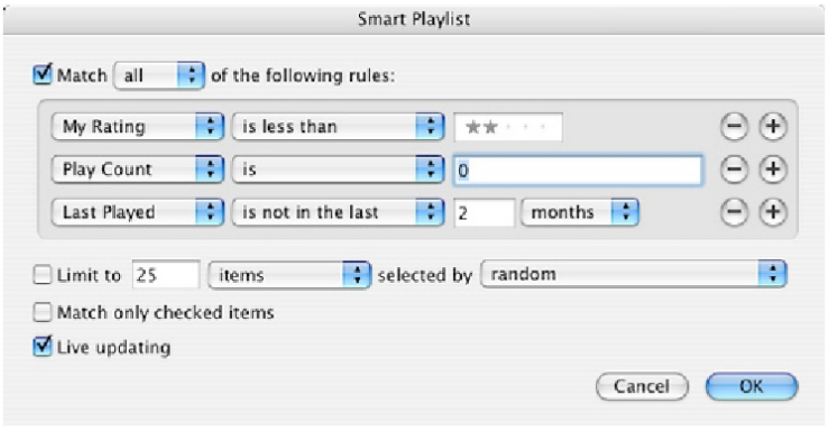

Figure 2.6: iTunes Smart Playlist[FL10].

specify a set of tags that match his desired music style to influence the playlist, these tags are used to select auto-tagged audio files.

As concluded earlier, emotions is an important area for a DJ as a part of their job is to understand them and shape them. A number of different views on the subject exist, e.g. emotions from a layman's point of view "emotions are what people say they are" and also the opposing views with a more synthetic approach [Sch05]. It is clarified that the difference between emotion and mood is that the latter is not so much linked to a specific event and also tends to be lower in intensity and spanning longer. Subjective feelings can be described as a three dimensional vector using the axes valence(positive-negative), arousal(calm-excited), and tension(tense-relaxed). The interaction aspect of emotion is also looked at through an evaluation of attitudes where it is found that this contains an affective component, a behavioral component, and a cognitive component. An example of an emotion-centered system is presented in [YIK11] as a system that takes emotion requests where the user can specify adjectives and their degrees. Each song is labeled through their proprietary "Acustic - Emotion" model which maps emotions to music attributes. Another emotion-centered example is the Mood Agent application, seen in figure 2.7.

HP DJ [Cli00] is a system which allows a user to specify a selection of tracks and with optional quantitative criteria. The system analyses the songs to see which ones that provide the best transitions to another track and use this in combination with the user criteria to perform sequencing and mixing. The user providable criteria include being able to specify the desired evolution of the tempo and being able to set specific rules such as "don't play Track A before Track B". The system also provides beat matching for improved cross fading. There is no no real time information used to adapt the playlist.

[BGA+08] presents a context-aware music player as an example of their proprietary WhozThat protocol. They have developed a piece of mobile software which allows for device identification over Bluetooth, this mobile software works by contiguously scanning the surroundings for a WhozThat music server. On connection the server will retrieve the users' favorite bands via the artists they have liked on Facebook. After this they use Last.FM API to generate a playlist with the most popular tracks for each artist and this playlist is then played.

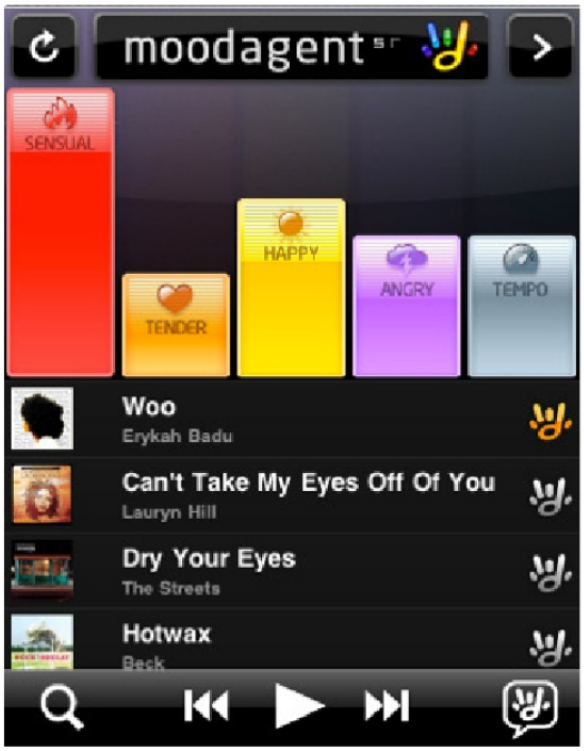

Figure 2.7: Mood Agent. The user can drag the five mood sliders to specify desired music[FL10].

2.7 Adaptive Playlist Systems

The previous section evaluated systems dealing with music recommendation based on a static set of input. For AutoDJ to work it needs to be an adaptable system meaning it needs to rely on techniques from "2.4 Ambient Intelligence" and "2.5 Adaptive Systems". This section will look at music systems which fall in the category of being able to adapt their playlists on the fly based on real-time input.

[PPW05] presents a system which is able to generate an initial playlist based on a seed song and then adapt it through user feedback in the form of skipping behavior. The user feedback is used to establish a set of heuristics (e.g. user is always skipping songs by a select artist). For the audio evaluation they use a similarity measure based on low-level audio features. They test four different playlist generation methods:

- (A)The n nearest neighbors to the seed song is used as a baseline list of song candidates.
- (B)The song candidate which is closest to the last accepted (ie. not skipped) is played as the next song.
- (C)The song candidate closest to any of the accepted songs is playedæ
- (D)The song candidate with the lowest distance any of the accepted songs and a (nearest accepted distance) / (nearest skipped distance) ratio below 1. If no such candidate, pick the one with the lowest ratio.

Their results shwo that (B) performs the worst, (A) performing second worst, (C) being second best and (A) being the best of the four. Their results show that the combination of audio similarity and simple heuristics is efficient at reducing the number of necessary skips. It should be noted that their testing was performed using hypothetical use case patterns and not on actual human users. A quality of their system is that it taps into the interaction the user already has with his music player.

[OKS06] has looked at how one can automatically generate a playlist based on physiological responses to music when combined with song meta data. Their system works in real time and relies on collaborative filtering. The idea with the system is that music can be used not only for enjoyment but also for a purpose. They mainly look at how this can help a user in terms of exercising.

Flytrap [CBH02] is a system which is built with the purpose of automatically creating soundtrack to please the majority of people in a room. Each potential user needs to be equipped with a Radio Frequency ID badge which contains his music preferences. These badges are used to detect when whom are in a room and the system then features an algorithm to find a compromise between the two most popular music genres. Flytrap fetches peoples' personal music taste by having software installed on their personal computer, looks at what they usually listen the most to. It has a built-in "DJ Taste" to e.g. never play the same artist twice in a row. While they do succeed in building a working system it appears to require quite a lot both in terms of both hardware and software.

Jukola[OLJ+04], see figure 2.8, is a jukebox system which allows people to upload more music over the internet.The system presents the usual gap between bar controlled hifi on the one hand and then a user controlled jukebox. It gets metadata and album art from freedb.org and amazon.com and allows for voting via handheld platforms. A downside is the hardware requirement, as it needs PDAs as clients. A slightly more updated approach to handheld voting is PartyStrands[MST+06] which allows the user to vote via text messages, this product was only shut down shortly after its inception though.

Figure 2.8: Jukola jukebox system featuring stationary and mobile voting clients.

HP DJ 2[Cli06] builds upon system from [Cli00], of the same author. The new version presented adds to the previous one the addition of floorshow feedback. The feedback comes in form of a wrist-attached custom portable device which features an up-vote and a down-vote button. This allows members of the crowd to signal their pleasure or displeasure with the current song being played. The signal is transmitted over Bluetooth to the HPDJ server

which uses the data to alter the playlist.

2.8 Summary

There are plenty of great existing music players such as Winamp or Spotify for which it seems more appropriate to integrate with than to build custom playback component. Lots of research has been done in the area of creating playlists but few systems can shape playlists in real-time according to user input. Of the adaptable systems all seem to require user interaction and cumbersome hardware setup. Based on techniques from ambient intelligence it should be possible to create an adaptable playlist system which can work autonomously.

Chapter 3

Design

3.1 Introduction

Based on the research from the previous chapter it is found that a system which is to be useful and novel is to have the following features:

- Automatic real-time adjustment.
- Operate entirely autonomously, i.e. without any interaction (implicit learning).
- Easy setup.
- Run on cheap commodity hardware.
- Work without an existing music library, i.e. needs to use a streaming music library.
- Integrate smoothly with an existing music player.

There are many different goals that one could aim for, but to keep things specific the goal of the system will be to maximize activity on the dance floor. This should be matter of the amount of people dancing but preferably the system should also take into account if the people are actively dancing or just standing and talking. The monitoring should be of the dance floor only and featuring no sensors for the rest of the venue. One could argue that by this decision cuts the opportunity of knowing if 20 people dancing is good (as it would be if there was 20 people in total) or bad (if the venue holds 200 other people not dancing). It is a clear limitation of the program but needed to keep things within a reasonable scope.

The focus is entirely on an adaptive playlist system, that being a system which can make an intelligent choice on the next song to play. Aspects such as mixing, scratching, fading, beat matching, or any other kind of audio manipulation will not be a part of the prototype.

The version of AutoDJ developed in this project will only be a very early and limited prototype. To accommodate for future versions it is important for the system to be designed in accordance with software engineering concepts such as MVC and loose coupling. This combined with a focus on modularity will make the system extendable without requiring a total rewrite. Modularity would allow further sensors to be added, anpther music player to be integrated, etc. without interferring with the other parts of the system.

A user interface is not specifically required for any of the core functionality. It does however seem relevant to provide some kind of interface to ease the debugging process. For debugging purposes the program should also have a learning mode where everything in the program is running except for the last link to the music player. This will allow one to test functionality through the console outputs without affecting the actual music playback.

There appear to be three steps in designing a system for automatic DJ'ing: Surveying the party, building a crowd profile, and finally playing back the music. These steps will shape the structure for the remaining sections of this chapter.

3.2 Step 1: Acquiring Live Audience Preferences

As found in "2 Previous Work" there are multiple ways that one can monitor a venue or get input from an audience. The core thing the system needs to sense can be expressed as *"How much does the current track help to keep people on the dancefloor"*. The requirement of no interaction rules out any kind of user carried device along with voting stations. The requirement of cheap commodity hardware rules out having to integrate sensors in the floor. Microphone input is ruled out due to a low presumed chance of getting the desired feedback. Smartphones have plenty of movement sensors such as accelerometer, compass, and gyroscope. To get data from these it would require an app to be installed, which would require user work. This kind of manual setup is not within the design goals and a smartphone approach would furthermore require a good amount of bandwidth and battery from the phones.

Computer vision is chosen as this provides essential feedback (dance floor activity) and a laptop with a webcam is commonly available. Most importantly, this kind of input can operate without user interaction and thereby work entirely autonomously. This section will look at how computer vision could work for this project and what possibilities there are in this area.

For motion detection the simplest algorithm is the two frame difference count. This simply adds up the pixel changes from one frame to the next. It is also possible to create a continuously updated model of the background and compare up against this one. This background model is typically an average of some hundred frames. Static images may also be used as background but these are very sensitive to any kind of environment changes or camera movement. To search for a specific image in another image one can use template matching [MHK06]. This works best only if one is searching for a very specific and unique item though, something which is not the case with party goers.

Detection of BLOBs (Binary Large OBjects) is typically done when one wants to identify the different parts of an image. This can be used to count individual parts of an image and provide grounds for classification. Testing of a few algorithms with this shows no promising results. A Kalman filter can be used to estimate the position, trajectory and speed of a moving object. In is in particular useful when combined with optical flow techniques together with which they also provide automatic background subtraction. This is a fairly complex algorithm and should not be attempted unless the previously described and more straight forward approaches turn out to fail.

Quick testing shows that simple motion detection algorithms provide promising results and it is therefore decided that the final implementation should work with that as a base.

3.3 Step 2: Selecting Next Track

The system needs to be able to pick the next track as the track that will deliver the most dance floor activity. To be able to make this prediction the system must first have a knowledge base of the audience preferences. This can be done by monitoring the venue as previously described and then based on this creating a list of played tracks along with their respective amount of dance floor activity. There are many independent factors such as the intoxication level of the party attendees or the general light level which could influence the motion level. With these independent factors changing over the course of a party it would make sense to introduce an adjusted motion level which more closely should map the implicit rating of the audience. This could be done by having all motion be compared to the average motion over e.g. the last 15 minutes and thereby create a more relative measurement.

To select a track one needs to know properties of any given track and it is found that online meta data sites such as Last.FM and Echonest provide web services with extensive meta data on tracks. There are many different properties that one might use to evaluate a track based on such as genre, artist, tempo, etc. The very most suitable appears to be similar tracks as that one already takes into account all the other properties. The amount of tracks one can find as similar via Last.FM differs a lot from track to track but it is usually around a handful. The similar tracks property is generated by Last.FM as a combination of many different methods including both content-based filtering and collaborative filtering which gives it a high degree of accuracy.

The proposed algorithm is to start out by trying to select the top similar hit to the top popular track. If this has already been played recently or is not found it should proceed with alternating between trying the next similar and trying the next popular track.

A step by step example of how this algorithm would work out:

- 1st similar to 1st popular
- 2nd similar to 1st popular
- 1st similar to 2nd popular
- 3rd similar to 1st popular
- 2nd similar to 2nd popular
- 1st similar to 3rd popular
- Etc.

If a song has been played recently it should not be played again for a certain period of time. In case no suitable track can be found for playback, e.g. because of a too little data collected, there should be a fallback mechanism which can put on just some music to prevent the system from going entirely mute. There are many other rules which could be applied but the ones

described here are considered the most important ones. In later iterations there are several other playlist factors which could be applied to improve the decision making.

If a track receives really poor feedback it appears necessary to evaluate if it needs to be cut. This raises the dilemma between that one wants to get rid of a song, that nobody wants, as soon as possible but cutting off tracks also disturbs the rhythm. The feature should be there but with a very low threshold to ensure that this only happens when really needed, the exact number needs to be tweaked through live testing.

3.4 Step 3: Music Playback

The system design goal is for it to be cheap and easy to setup, this means that the user should not bother with assembling a music collection himself and a streaming service should therefore be used. As found in the previous work section, there are multiple streaming radios which are potential products to integrate with in terms of handling the playback component. Pandora, while being the original provider of music recommendation, is today only available to users in the United States, which excludes it from this project. Grooveshark has been accused to right issues and is deemed illegal in Denmark. Last.FM excels in terms of an extensive metadata library but only has a web-based player.

Spotify is very popular, has the largest collection and mostly importantly it features a desktop client, which should ease integration from other programs. Spotify will therefore be the choice of music player to integrate with.

Chapter 4

Implementation

4.1 Introduction

With a design in place it is possible to proceed to the development of the AutoDJ system. This chapter will describe the technical considerations and details which surfaced in the attempt to implement the design. The solution has been implemented in C#. This language was picked as it features lots of high quality frameworks in areas such as computer vision and web service interaction. A downside is that the language is executed via virtual machine which means that it generally is a bit slower than C++. Should this be an issue, it should be possible to optimize the program using GPU calculations.

There are many uncertainties in terms of how much of the design is actually technically possible. For use in high-uncertainty projects one can benefit from using Scrum, an agile methodology that operates via iterations[Sch09]. These iterations can be planned from a high-level perspective but the actual implementation tasks will be defined between each sprint. This is done by keeping a constant backlog of user stories for the product, of which the ones deemed most important are selected for each sprint. For this project three sprints can be defined: First, Spotify integration should be attempted as being able to play music, even in a naive way, would provide a basic music player. Secondly computer vision monitoring should be implemented to see which data one can get there. Finally a 'brain' can be built as a component that can learn from these computer vision data.

To ease the reading flow, the order of presentation here will follow the natural data flow which also falls in line with the structure in the design chapter. Unless otherwise noted the implementation follows the design specified in the previous chapter.

All code can be found on the accompanied CD along with a VS project. An overview of all classes can be found in the Appendix.

4.2 Step 1: Computer Vision

AForge.NET[1] was picked as a computer vision library as it is an open source framework for designers and researchers working in the fields of computer vision and artificial intelligence. The framework holds multiple libraries on image processing, computer vision, neural networks, and machine learning. For this project it is primarily the vision library that is used as that one features components for motion detection.

The webcam used is a Logitech C170[2], a 15eur webcam, which testing shows to be of a sufficient quality. It runs 720p at 25fps. Both two-frame difference and background modeling are implemented and tested. Testing shows that they work equally great as indicators of motion level.

Data collection is performed every frame and every second it is then summarized and used by the brain class. The brain class and all methods in it are run on a 1sec update tick. Data from all frames between update ticks are summarized each tick. This approach has the benefit of low CPU usage while also maintaining all frame input data. Each tick the average for the last 25 frames and for the last 600 seconds is found. These are used to create an implicit audience rating based partly on the relative change between the last 25 frames and the last 600 seconds. The longer average uses the mean motion level for each second as that one is already available and this allows for increased performance.

4.3 Step 2: Track Selection

In each tick, after the averaging, a check is made if the current song is done playing. There is no direct way of querying this from Spotify and instead a small method is used which compares the time into each track with the track length and uses this to determine if the track will be done playing within a few seconds. If a track is considered done playing the search for the next track to play begins. As implemented now it starts playing the next track right away, a better solution would probably be to start the search earlier and then delay the actual playback until just at the last second. This has been kept simple for now as this part is likely to be completely rewritten when features like fading are to be implemented.

The search for the next track to play requires a lookup to the *KnowledgeBase* class which holds the ratings of all tracks played so far. From this class the top tracks are picked as per the algorithm specified in the design and as seen in listing 4.1.

```
while (similarSongId == null)
{
  while (popularityIndex < SimilarityLimit && popularityIndex < Tracks.
      ContentList.Count)
          {
    if (popularityIndex == popularityLimit)
    {
      similarityIndex = popularityIndex + 1;
      popularityIndex = 0;
      popularityLimit++;
```

[1] https://code.google.com/p/aforge/
[2] http://www.logitech.com/en-gb/product/webcam-c170

```
    }
    else
    {
      popularityIndex++;
      similarityIndex--;
    }
    similarSongId = GetSimilarTrackURI(popularityIndex, similarityIndex);
  }
}
```

Listing 4.1: Code for determining the next track to play.

The *GetSimilarTrackUI* method leads to a method inside the *MusicInfo* class which will handle the actual lookup of a similar track. Having settled on Spotify as the music player it would make sense to also use their APIs for meta data extraction, the Spotify API does however not allow for query of the track similarity and instead the Last.FM api is used for this. Echonest was also considered, as they are one of the largest music meta data providers, but they do not provide similarity lookup on a track level.

To gain access to the Last.FM web service one needs an API account which is free for non-commercial use. With a such one acquired one needs to pass along an API account key for all requests. The Last.FM API is a web service which can be accessed via a HTTP GET lookup that returns a JSON(JavaScript Object Notation) or XML(Extensible Markup Language) string which can then be deserialized [3]. JSON was picked over XML as it requires a smaller data size and is also faster to parse due to it just being just key/value pairs which are easily mappable[Cro06].

The JSON format is a structure for data exchange which is easy to read for humans and also easily parsed by machines. It is language independent and built on two structures: JSON Object and JSON Array. A JSON object is a collection of name/value pairs while a JSON array is an ordered list of values (i.e. very similar to a regular array in most languages). Getting data from the JSON therefore requires one to parse JSON, this is accomplished with the help of the Json.NET library[4]. The most important lines of code for the JSON parsing can be seen in listing 4.2.

```
string jsonString = WebHelper.HttpGet(url);
string artistName = (string)json["similartracks"]["track"][index]["artist"][
    "name"];
JObject json = JObject.Parse(jsonString);
string trackName = (string)json["similartracks"]["track"][index]["name"];
```

Listing 4.2: Parsing JSON data via LINQ queries in JSON.NET in order to get artist and track name.

To keep data transfer reasonable, the amount of tracks to lookup has been limited to the five most popular. This number could have been dynamic but is fixed to allow for caching. As a fallback measure the system will resort to a playlist of the most popular dance tracks in Denmark in case it can not settle on a track to play next.

[3] The process of converting from a string of characters to data objects

[4] `http://json.codeplex.com`

4.4 Step 3: Music Playback

Multiple different attempts at acquiring control over Spotify playback were made. One of the entry points Spotify provide is "Spotify Apps" which are small code pieces that can run from within Spotify. This however would impose a number of restrictions such as only being able to use HTML5 and also a such route would very much lock AutoDJ to Spotify. Another option would be to make use of "libspotify" which is an official C API package, this one however requires a Spotify Premium account and is therefore ruled out. In the end it was found that the most feasible solution was to make use of the entry points that are provided for websites.

The chosen integration was done via a Spotify process running called SpotifyWebHelper.exe which runs in the background as long as Spotify is running. It works by listening on port 4380 and is normally used for the play buttons on Facebook and embedded players on other websites. There is an API running on that port, providing several functions that can be used with either JSON or XML. Using the SpotifyLocal[5] framework it is possible to hook up to this API. This way of integrating ensures that even if AutoDJ crashes it is nearly impossible for it to take down Spotify (and thereby the music playing) with it.

Everything in Spotify can be accessed via URIs(Uniform Resource Indicators) such as "spotify:track:ID" for a single track or "spotify:user:sonymusicdenmark:playlist:ID" for a complete playlist. With the desired music track coming from Last.FM a conversion from the "Artist - Track" naming scheme to Spotify URI is needed. This was handled via the Spotify Search web service, their services take arguments via HTTP get and can return in either JSON or XML. Just as for the *MusicInfo* class, JSON is also picked as format of choice here.

4.5 General

A UI was implemented using Windows Forms which is a classic framework for .NET UI development. The UI is handled on a separate thread which ensures that it should be responsive at all time. The rest of the program is implemented on a single synchronous thread which means that the code is waiting for e.g. web lookups to return. While this does not appear to be any issue (as the lookup usually happen in less than a second anyway) then it would still be better system design to let handle improve the handling of this. Performance shows that the application uses less than 10 percent of the CPU on a 2.2 Ghz mobile Ivy Bridge meaning it should run without issues on any laptop.

All web lookups were directed over the *WebHelper* class which was built to provide simple caching. This means that whenever a lookup is performed which has been requested before, it will instead return the response from last time. No expiration or cache clearing functionality have been added as the responses are assumed static for the duration of the program. The cache is entirely in-memory (on the heap) meaning it is very fast to look up. The addition of caching made the program more responsive and also limits the risk of overusing the web services[6].

[5] `http://code.google.com/p/spotify-local-api/`

[6] Both Spotify and Last.FM allow only a limited amount of requests per second

A diagram of the system architecture showing data flow can be seen in figure 4.1.

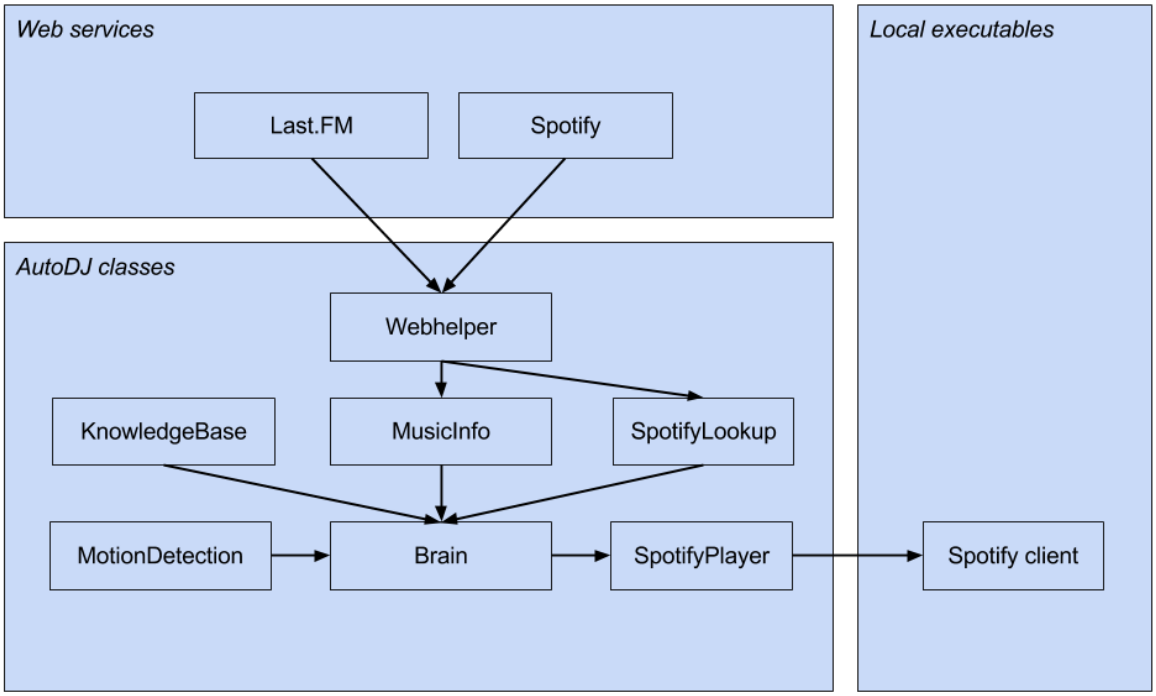

Figure 4.1: An architectural overview of the most important entities in the system and the data flow between them. Hardware and framework use is not listed for the sake of simplicity.

Chapter 5

Test

5.1 Method

During implementation many small scale tests were carried out but they were all of technical character and executed outside the intended environment for the system. With a rough prototype implemented it is time to see how well it holds up when presented with a realistic scenario of use. This test should gather data on the overall feasibility of the system and provide input for further revisions.

A user test was conducted at the university Friday bar at Aalborg University Copenhagen. The venue is student driven and holds Friday bars every two to three weeks for all students at the campus. The opening time is from 1500-2200h and usually there is bar managed playlist playing from 1500-1800h and then from 1800h a professional DJ takes over for the remaining hours. The AutoDJ system was on this day running during all hours. The bar manager knew of the system but the rest of the attendees did not know that there would be anything different about the DJ for the event. The setup was placed in a corner of the room next to the dancefloor with the webcam at a high table.

Test data are gathered in three ways:

- Quantitative measurements of the dance floor rating and songs being played.
- Interviews with people who have been out dancing.
- Observation.

The benefit of using three different data sources is that one can triangulate between them which helps improve the validity of the results. Questionnaires were considered but it was deemed more likely that party goers would want to be interviewed than wanting to fill out surveys. Interviews furthermore have the benefit of one being able to ask follow up questions in case the subject seems to have misunderstood a question or if other interesting points comes up. A downside of talking with people in general at a party venue is that they are likely to have consumed alcohol which can impact their ability to reason.

The intention with the quantitative part is to get some objective data to compare up against. The goal with the observation is to get a larger understanding of how the quantitative log

data and the qualitative interview fit together. The interview part focused on two central questions: Why they had started dancing and why they had stopped dancing. This was noted along with a time stamp, age, gender and any other thoughts they had on the music of the night. For the interviews I presented myself as the DJ and as doing research on "dance culture".

All log and interview data can be found on the CD.

5.2 Findings

A total of 21 people were interviewed, 12 males and 9 females. Aged between the age of 20 and 29 with a mean age of 23,71 (std. dev 2,28). There was around 30-35 people in total at the venue, this number being relatively stable between 1600h and 2200h.

The system was kept in learning mode until 1930h as there had been no dance floor activity which it could learn from. At that point it was decided to enable it anyway and see how things would workout. Until 1956h there had been no dancefloor activity, so at this point a requested song was put on which enabled the first handful to come on the dance floor. Taking requests was not a part of the plan but was accepted in an attempt to get any kind of dance floor activity.

A Limbo event was initiated by the Bar managers lasting between 2128h and 2140h. During this time frame AutoDJ in learning mode and music was controlled manually. Disco lights were turned on at 2045h.

A screenshot of the program in action during the test can be seen in figure 5.1.

5.2.1 Log Data

For quantitative evaluation data was logged each second for around 5 hours of data equivalent to roughly 20.000 data points. For each time stamp, motion level was logged. Also logged was which song that was being put on at which time and when the system had was unable to find a suitable next track. Manual playbacks in Spotify were only logged with time stamp and not which song that was put on.

The system logged 18 fallback attempts, i.e. times where it could not find a suitable track to play next. The last of these fallback attempts happened at 1748 meaning none of them happened after the system was taken out of learning mode. Seven times during the test a song was put on manually in response to public request.

An overview of the measured dance floor activity throughout the evening can be seen in figure 5.2. The graph starts at 1900h, as there were no people dancing for the first several hours. The massive fluctuations are due to this being raw data, actual dance floor activity was significantly less fluctuating. The graph generally shows that there were no dance floor activity until around 2000h, after which there were varying amounts of people dancing for the remainder of the night.

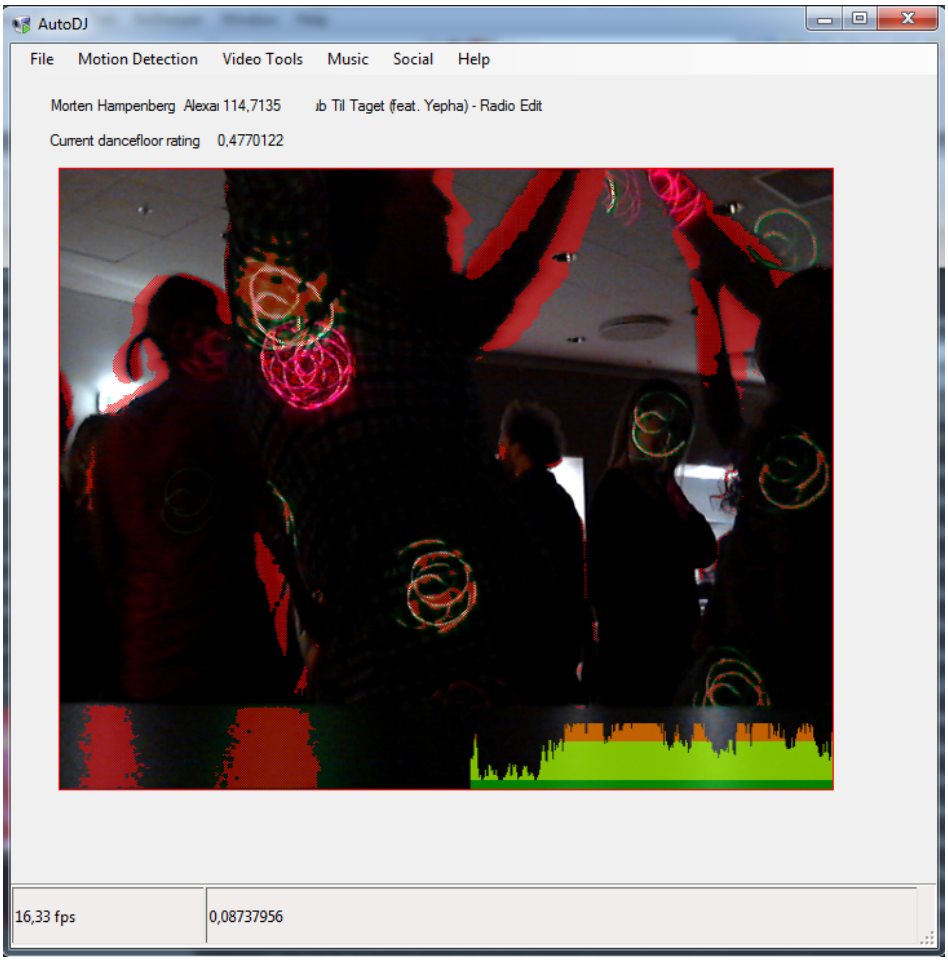

Figure 5.1: The final AutoDJ prototype in function at the test venue. Red spots denote motion and the bar the button right shows motion history over 300 frames. The top right text shows the currently highest rated track along with its rating.

5.2.2 Interviews

Examples of answers to why people entered the dancefloor range include *"It was a total rad song that came on"*, *"There were hot girls on the dance floor"*, and *"There was a nice hiphop beat going"*. Examples of reasons for leaving the dancefloor include *"Lack of other people dancing"*, *"Was sweating too much"*, and just plain *"Got tired"*. 13 of the 21 respondents mentioned a music related reason for entering the dance floor and thereby makes music the most prominent parameter in terms of attracting dance floor activity. Only two respondents mentioned a music related reason for leaving the dance floor.

When asked about their thoughts on the music being played there were no significant complaints or comments about the quality of the DJ which leaves the impression that the system was indeed mistaken for providing a real DJ experience. There were however a comment about the music not being fresh enough and relying too much on well known hits.

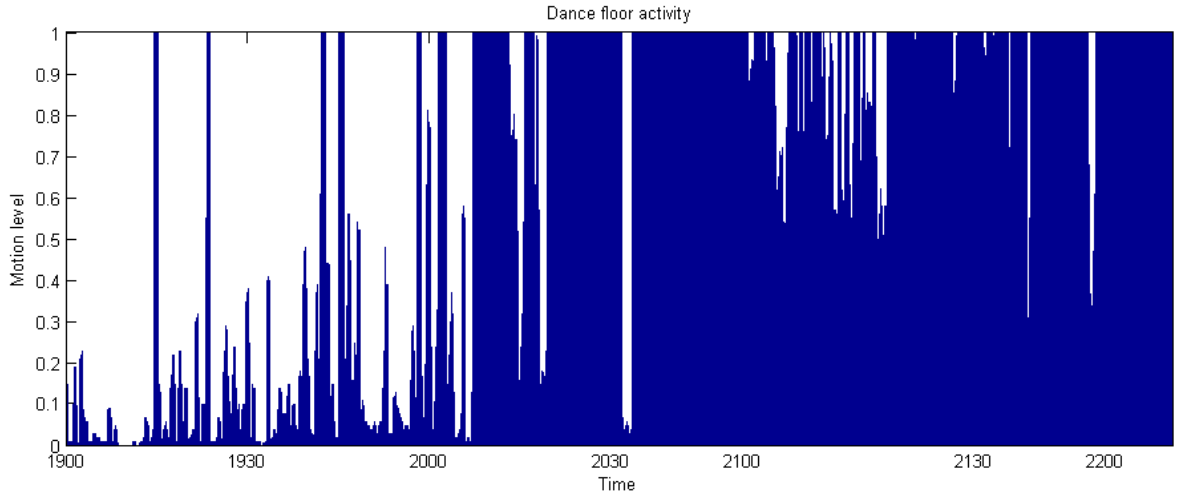

Figure 5.2: Motion level history as recorded by AutoDJ at the test venue in the timespan from 1900h to 2200h. The highest data points are capped at 1 as this point means the dance floor is filled and values above that are too fluctuating. The x axis is slightly stretched at times due to the program crashing a two times meaning there is data missing in some places.

5.2.3 Observation

There were too few people in total at the venue to create a proper atmosphere for dancing. This somewhat changed over the evening though, for the last hour the atmosphere ended up being so good that there were people on the dancefloor constantly.

Through continuous observation of the activity readouts it was clear that it matched with the actual amount of activity on the dance floor. Tracking when it was darker still work ok, there were just lower ratings but due to the relative design of the system this was not an issue. The disco lights coming on at 2045h did not seem to have any noticeably impact on tracking performance either.

At first the system was too aggressive with cutting off unpopular tracks so this had to be tweaked. Having a cut off proved valuable in a situation where the dance floor was filled with people dancing to a song and by the ending of that song a somewhat less popular song came on. The dancing stopped and multiple people started yelling *"we want a new track"*. Before I could think of what to do, the system had already picked up on this desire and instead put on another track which fell in the taste of the audience and the party continued at full power.

At the end of the Friday Bar the system faced the most criticism when the music had to be turned off. At that point the atmosphere was really good and the system had gotten a good hang of what people wanted to hear meaning that the dancers were facing a very sudden stop in the music. At the end of the party the bar managers requested for the system to be used for the next Friday bar also. It is uncertain how much this is a praise of the music selection or just economic reasoning based on that they had gotten a DJ for free unlike their regular paid DJ.

5.3 Discussion of Results

There were a number error sources which influenced the test results and multiple things which could be improved, these matters which will be discussed here. Generally the technical parts of the program worked out well. There were a two crashes but neither of them seemed to get noticed by the crowd (probably due to them never interrupting the music). The one crash was due to illegal characters in a set of meta data from Last.FM while the other crash was of unknown origin.

Due to the horizontal angle of the camera there was a strong distance bias, i.e. people closer to the camera had more impact in motion measurements. Ideally the webcam should have been placed in the ceiling and equipped with a wide lens.

A missing part of the data logger is that it does not catch when music tracks are being played manually in Spotify, such as the case with requests. It is therefore unsure at which points song were played manually. The graph in figure 5.2 does not match entirely with the observed development of the dance floor activity as it shows decreasing activity around 2100h, which is not in line with observations. This might be due to the sun setting down around that point which brought a darker room.

Only deactivating AutoDJ to learning mode during Limbo and not entirely shutting it off was a mistake as motion levels during that time frame were more related to the Limbo game going on than the music itself.

Several of the interview subjects were noticeably intoxicated which had a negative impact on their ability to formulate meaningful answers. Even with multiple rephrasings of the question, one subject insisted that the one and only reason for him entering the dance floor was that *"he was Swedish"*. Most interview subjects seemed to have interpreted me as also being the DJ of the night, this might have had an unfortunate influence on their willingness to criticize music choices.

This test did not uncover how the system holds up against a real DJ or a playlist as that would take a proper control scenario. A test for the next iteration of the system would be to alternate every half an hour between having AutoDJ active and just playing a playlist of popular dance tracks.

Chapter 6

Discussion

After having been through research, design, implementation, and test it is now possible to take a broader view on the project as a whole and reflect upon the journey that has been taken. Lots of problems were tackled, many of which seemed impossible to overcome in the beginning. Although great strides were made, so were some mistakes and the final prototype definitely has its shortcomings. The initial idea with the AutoDJ system was to make predictions on what music people would like to hear next when dancing. Already by the word prediction one can see that one is dealing with an area where a clear answer is no straightforward or easy thing to provide. Or maybe it is easier than made out to be, maybe popular tracks are just popular with almost any crowd of party goers and thereby the crowd-adjusting features are less meaningful.

A conclusion from the test is that the system currently has a clear cold start issue and requires some manual curation to gain an initial knowledge base. To take care of this some kind of starting medley or random selection could be made. Another interesting option would also be to let people log in somewhere via their Facebook or Spotify profiles and then this could serve as a seed music preference.

The test also showed the plain verbal requests from the audience as a fairly effective way of getting people on the dancefloor. This goes to say that instead of trying to predict things with advanced technology, one might simply be able to ask people. Song requests were very much something people seemed to want so it would make sense to integrate in the system. An extension to AutoDJ would be to make use of the wide smartphone availability today and let users request songs through them.

In terms of adaptability the test showed a conflict at the early hours of the test between system wanting people to dance and people not wanting to dance at early hours. This was a natural result of the system only featuring the goal of maximizing dancefloor activity. The system could benefit from having settings to give some rough indicators of when it should go for lounge music and when it should go for dance music. Similarly, the final minutes of the test clearly showed that the system needs a cool down feature as the one described in [Cli06].

Chapter 7

Conclusion

This report has presented the design and implementation of the AutoDJ system. It is an adaptive music player that can adjust the music to be played based on input from computer vision and by using 3rd party web services. The system fulfills a number of design criteria such as being able to operate with commodity hardware and without any kind of user interaction.

The system was tested with users at a party venue which overall delivered satisfactory results. The test showed that the technical parts worked out well and that the system performed the needed task even with many components relying on simple solutions. The users were in general appreciative of the music selection but it is not possible to conclude on exactly how the system compares to a static playlist or to a professional DJ.

The system involves so many different problem areas, most of which it has only been possible to scratch the surface of during this project. Entire projects could easily be devoted to improving each component such as the computer vision system, the track selection algorithm, the similarity system, and the many other components that power AutoDJ. This project focused on motion level activity but it would make sense to look at if people counting, facial expressions, dance style, or other features could provide a more accurate picture of the atmosphere of a venue. Entirely new options opens up if this system was implemented in a sufficient number of bars and discos around town it would open up for a lot of opportunities. Connecting all of the systems to an online server one would be able to look up a bar before going there to check for things such as *"what kind of music are they playing tonight?"* or *"are there lots of people on the dance floor?"*. It would also provide venue owners with lots of possibilities in terms of business intelligence.

Bibliography

[BB00] Bill Brewster and Frank Broughton. *Last night a DJ saved my life: The history of the disc jockey.* Grove Press, 2000.

[BGA+08] Aaron Beach, Mike Gartrell, Sirisha Akkala, Jack Elston, John Kelley, Keisuke Nishimoto, Baishakhi Ray, Sergei Razgulin, Karthik Sundaresan, Bonnie Surendar, et al. Whozthat? evolving an ecosystem for context-aware mobile social networks. *Network, IEEE*, 22(4):50–55, 2008.

[BMEM10] Thierry Bertin-Mahieux, Douglas Eck, and Michael Mandel. Automatic tagging of audio: The state-of-the-art. *Machine Audition: Principles, Algorithms and Systems. IGI Publishing*, 2010.

[BMEWL11] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*, pages 591–596. University of Miami, 2011.

[CBH02] Andrew Crossen, Jay Budzik, and Kristian J Hammond. Flytrap: intelligent group music recommendation. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 184–185. ACM, 2002.

[CHR+00] Antonio Camurri, Shuji Hashimoto, Matteo Ricchetti, Andrea Ricci, Kenji Suzuki, Riccardo Trocca, and Gualtiero Volpe. Eyesweb: Toward gesture and affect recognition in interactive dance and music systems. *Computer Music Journal*, 24(1):57–69, 2000.

[Cli00] Dave Cliff. Hang the dj: Automatic sequencing and seamless mixing of dance-music tracks. *HP Laboratories Technical Report HPL*, (104), 2000.

[Cli06] Dave Cliff. hpdj: An automated dj with floorshow feedback. In *Consuming Music Together*, pages 241–264. Springer, 2006.

[Cro06] D. Crockford. The application/json media type for javascript object notation (json). *Network Working Group*, 2006.

[Den01] Peter J Denning. *The Invisible future: the seamless integration of technology into everyday life.* McGraw-Hill, Inc., 2001.

[DMV97] AM De Mooij and WFJ Verhaegh. Learning preferences for music playlists. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[FL10] Ben Fields and Paul Lamere. Finding a path through the jukebox: the playlist tutorial. In *11th International Society of Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[Gol03] Paul Goldstein. *Copyright's highway: From Gutenberg to the celestial jukebox.* Stanford Law & Politics, 2003.

[GWS+06] William T Glaser, Timothy B Westergren, Jeffrey P Stearns, Jonathan M Kraft, et al. Consumer item matching method and system, February 21 2006. US Patent 7,003,515.

[HML09] Michael D Harrison, Mieke Massink, and Diego Latella. Engineering crowd interaction within smart environments. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 117–122. ACM, 2009.

[Hor96] Nick Hornby. *High Fidelity.* Riverhead, 1996.

[LR08] KuanTing Liu and Roger Andersson Reimer. Social playlist: enabling touch points and enriching ongoing relationships through collaborative mobile music listening. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pages 403–406. ACM, 2008.

[LSST] Daniel Lew, Ben Sowell, Leah E. Steinberg, and Amrit S. Tuladhar. Recommendation systems: General collaborative filtering algorithm ideas.

[LVH05] Tuck W Leong, Frank Vetere, and Steve Howard. The serendipity shuffle. In *Proceedings of the 17th Australia conference on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future*, pages 1–4. Computer-Human Interaction Special Interest Group (CHISIG) of Australia, 2005.

[LVH06] Tuck Wah Leong, Frank Vetere, and Steve Howard. Randomness as a resource for design. In *Proceedings of the 6th conference on Designing Interactive systems*, pages 132–139. ACM, 2006.

[MCM86] Ryszard S Michalski, Jaime Guillermo Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann, 1986.

[MED+09] François Maillet, Douglas Eck, Guillaume Desjardins, Paul Lamere, et al. Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[MHK06] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.

[MLDLR03] M. Montaner, B. López, and J.L. De La Rosa. A taxonomy of recommender agents on the internet. *Artificial intelligence review*, 19(4):285–330, 2003.

[MST+06] Francisco J Martin, Jim Shur, Marc Torrens, Rick Hangartner, Guillermo Caudeville-Laliena, David Del Ser Bartolome, and Craig Rowley. Dynamic interactive entertainment, August 31 2006. US Patent App. 12/278,148.

[OKS06] Nuria Oliver and Lucas Kreger-Stickles. Papa: Physiology and purpose-aware automatic playlist generation. In *Proc. 7th Int. Conf. Music Inf. Retrieval*, pages 250–253, 2006.

[OLJ+04] Kenton O'Hara, Matthew Lipson, Marcel Jansen, Axel Unger, Huw Jeffries, and Peter Macer. Jukola: democratic music choice in a public space. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 145–154. ACM, 2004.

[Pou13] Jacob ScratchMagic Poulsen. Private interview, April 8th 2013.

[PPW05] Elias Pampalk, Tim Pohle, and Gerhard Widmer. Dynamic playlist generation based on skipping behavior. In *Proc. of the 6th ISMIR Conference*, volume 2, pages 634–637, 2005.

[Sch05] Klaus R Scherer. What are emotions? and how can they be measured? *Social science information*, 44(4):695–729, 2005.

[Sch09] K. Schwaber. *Agile project management with Scrum*. Microsoft Press, 2009.

[SKKR00] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167, 2000.

[Web01] William Weber. From miscellany to homogeneity in concert programming. *Poetics*, 29(2):125–134, 2001.

[YIK11] Ryosuke Yamanishi, Yuya Ito, and Shohei Kato. Automated song selection system complying with emotional requests. In *Entertainment Computing–ICEC 2011*, pages 367–370. Springer, 2011.

Appendix

Class Diagrams

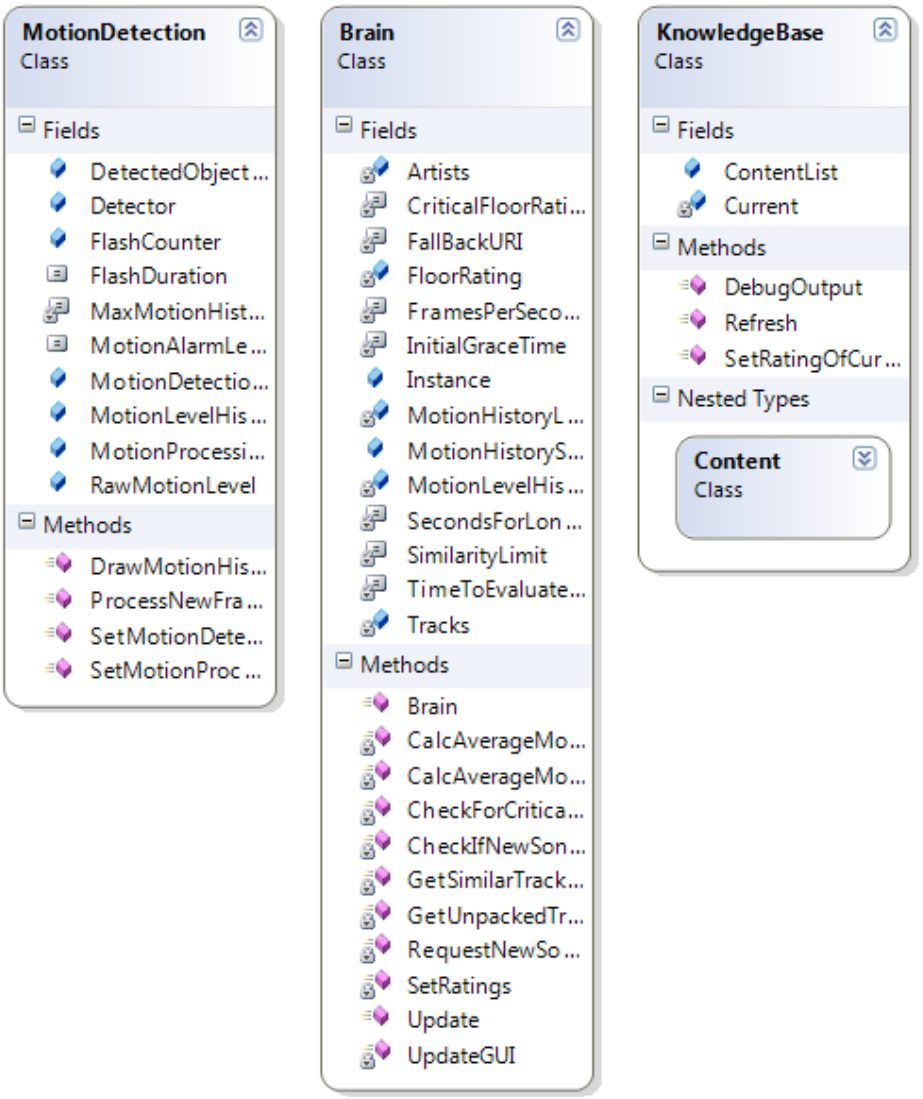

Figure 7.1: Classes in AutoDJ.

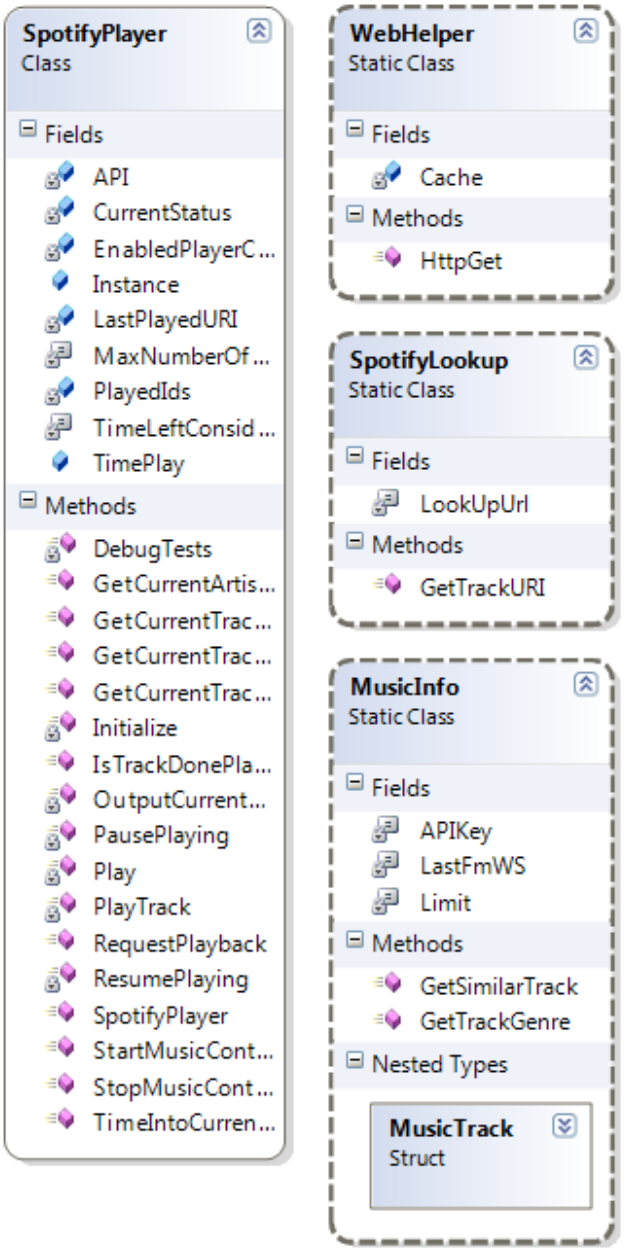

Figure 7.2: Classes in AutoDJ.