

**UNIVERZITA KOMENSKÉHO V BRATISLAVE  
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**



**VIZUALIZÁCIA HUDBY S INFORMAČNOU HODNOTOU**

**BRATISLAVA 2010**

**Bc. Lukáš Duchovič**

**UNIVERZITA KOMENSKÉHO V BRATISLAVE  
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**Vizualizácia hudby s informačnou hodnotou**

**Diplomová práca**

Študijný program : Aplikovaná informatika

Študijný odbor: Počítačová grafika a videnie

Školiace pracovisko: KAI

Školiteľ: RNDr. Stanislav Stanek PhD.

**Bratislava 2010**

**Bc. Lukáš Duchovič**

## ZADANIE DIPLOMOVEJ PRÁCE

Študent(ka) ..... Lukáš Duchovič .....

vypracuje za účelom obhajoby v rámci záverečného konania vo vysokoškolskom magisterskom štúdiu na UK FMFI diplomovú prácu s (predbežným) názvom:

..... Vizualizácia hudby s informačnou hľadiskom .....

pod vedením pracovníka ..... S. Stanek ..... z pracoviska KAI .....

Cieľ diplomovej práce a ďalšie poznámky:

Prehľad doteraz používaných techník (FFT, Beat detection) na vizualizácii hudby, ich implementácie a prezentácia ich výhod a nevýhod.

Experimentovanie s vizualizačnou hľadiskom, ktoré by priniesla aj hudobne využitelnú + podpis študenta Duchovič ..... podpis ved. dipl. práce Han

Diplomová práca je tematicky priradená k špecializácii Aplikovaná informatika

..... (Duchovič) .....  
V Bratislave dňa ..... podpis gestora špec.: ..... (Duchovič) .....

Poznámka: Vyplňujte v 3 exemplároch: 1 pre študenta, 1 pre vedúceho práce, 1 pre študijné oddelenie

(odovzdá študent pri zápise do 4. ročníka).

\* informáciu (tónina, takt, časy nástupov, charakteristické skladby - duri, mol). Súčasťou práce bude komunikácia s profesionálnymi hudobníkmi o ich predstave využitejnej hudobnej vizualizácie.

## **Čestné prehlásenie**

Čestne prehlasujem, že som túto prácu vypracoval samostatne s použitím uvedenej literatúry a informačných zdrojov.

Lukáš Duchovič

.....

## **Pod'akovanie**

Chcel by som pod'akovať môjmu vedúcemu Stanislavovi Stanekovi za ústretový prístup a za odbornú pomoc v kritických situáciach, ktoré nastali pri tvorbe tejto práce. Tiež d'akujem všetkým, ktorí ma podporovali, inšpirovali a ktorých som na úkor tejto práce musel ignorovať.

Lukáš Duchovič  
Bratislava, 2010

# **Vizualizácia hudby s informačnou hodnotou**

**Lukáš Duchovič**

Univerzita Komenského v Bratislave  
Fakulta Matematiky, fyziky a informatiky  
Katedra Aplikovanej Informatiky  
RNDr. Stanislav Stanek PhD.

## **Abstrakt**

Cieľom mojej práce bolo preskúmať vizualizáciu hudby v podobe aká je v dnešnej dobe dostupná. Predstaviť a implementovať techniky, ktoré sa v tomto odvetví používajú a prezentovať ich výhody a nevýhody. Prínosom práce je experimentovanie s novými technikami, ktoré môžu do vizualizácie priniesť aj skutočné informácie o hudbe. Súčasťou práce je aj prieskum s ľuďmi rôzneho hudobného vzdelania, ktorí prezentujú svoje predstavy o pre nich využiteľnej vizualizácii.

## **Kľúčové slová**

vizualizácia hudby, frekvenčná analýza, FFT, beat detection, echonest

# **Music visualization with information**

**by Lukáš Duchovič**

Comenius University in Bratislava

Faculty of Mathematics, Physics and Informatics

Applied Informatics

RNDr. Stanislav Stanek PhD.

## **Abstract**

The goal of this diploma thesis is to review music visualization in form as it is known in presence. We introduce and implement techniques, which are used to visualize music and review theirs pros and cons. The contribution of this work is in experiments with methods, which can bring real information about music. This work includes communication with people with different level of musical education, who presenting their opinions about usable and music visualization.

## **Keywords**

music visualization, frequency analysis, FFT, beat detection, echonest,

# Predhovor

Hudba, ako vieme, je jedno z umení, ktoré ľudia považujú za neoddeliteľnú súčasť ich životov. Ak by sme pátrali priamo po význame slova hudba, nájdeme množstvo definícií. Niektoré slovníky definujú hudbu ako zvuky, ktoré ľudia považujú za príjemné[1], iné zasa definujú hudbu ako umenie, ktorá kombinuje vokálne a inštrumentálne zvuky a využíva ich na vyjadrenie pocitov[2]. Tiež môžete nájsť slovník, v ktorom hudbu opisujú ako napísané, alebo vytlačené znaky, ktoré reprezentujú zvuk.[2]

Každopádne, v konečnom dôsledku je úplne jedno aké slová si ľudia vyberú na reprezentáciu slova hudba. Každá osoba v ľudskej populácii na tejto planéte rozumie tomuto slovu. Tak ako pozná hudbu podnikateľ v New Yorku, pozná ju aj domorodec z opusteného ostrova blízko Afriky. Samozrejme, každý z nich si predstavuje inú formu hudby, ale v podstate ide o ten istý fenomén.

Je to umenie, ktorého hlavným médium je zvuk. Medzi ďalšie významné premenné hudby patrí tón (jeho výška je dôležitá pri vytváraní melódií a harmónií), rytmus, a samotná charakteristika zvuku. Ak by sme chceli byť priveličmi dôslední, vieme povedať, že každý zvuk je v podstate hudba, aj keď ak by sme pokračovali v takejto diskusii, dostali by sme sa do filozofickej roviny, ktorá by nás donútila zapísať nespočetné množstvo stránok.

A o čo teda v téme zvanej vizualizácia hudby presnejšie ide? Vo svojej podstate to je jednoduché, ako som vyššie spomínal, hlavné médium hudby je zvuk. Mnoho ľudí sa však zamýšľalo, čo by sa stalo, keby človek dokázal zvuk transformovať na vizuálny vnem. Ako by vyzerala tá a tá hudba, dokázala by byť hudba vnímaná rovnako intenzívne aj bez slchu? Tieto otázky započali vývoj spôsobu, ako by mohla byť hudba zvizualizovaná aj pre iný zmysel ako je sluch.

Existuje veľké množstvo prístupov, väčšina z nich sa sústredí na vizuál a v podstate má byť len doplnkom k hudbe. Jedna z najpresnejších vizualizácií hudby

existuje už od pradávnych čias v podobe nôt, avšak táto vizualizácia je pre neštudovaného poslucháča absolútne nevyužiteľná.

V tejto práci je hlavným motívom zamyslenie sa nad vizualizáciou v podobe ako dnes funguje a používa sa. Aké ma výhody, nevýhody a použitie. Tiež sa zamýšľame ako by mohla vizualizácia hudby vyzerat v budúcnosti a ako by vôbec podľa ludí mala vyzerat? Vedeli by profesionálny hudobníci využiť druh vizualizácie, ktorá by obsahovala nejakú informáciu?

# Obsah

<b>Úvod</b>	<b>14</b>
<b>1. História</b>	<b>15</b>
<b>1.1 MilkDrop</b>	<b>17</b>
<b>1.2 G - Force</b>	<b>18</b>
<b>2. Zvuková analýza</b>	<b>20</b>
<b>2. 1 Úvod o zvuku</b>	<b>20</b>
<b>2. 2 Fourierova transformácia</b>	<b>23</b>
<b>2. 2. 1 Diskrétna Fourierova transformácia</b>	<b>24</b>
<b>2. 2. 2 Rýchla Fourierova transformácia</b>	<b>27</b>
<b>2. 2. 3 Goertzelov algoritmus</b>	<b>29</b>
<b>3. Beat detection</b>	<b>31</b>
<b>3. 1 Simple Beat Detection</b>	<b>32</b>
<b>3. 2 BTS</b>	<b>36</b>
<b>3. 2. 1 Popis systému BTS</b>	<b>39</b>
<b>3. 2. 2 Predpoved' beatu</b>	<b>43</b>
<b>3. 2. 3 Výsledky BTS</b>	<b>47</b>
<b>4. Informačná hodnota v hudbe (prieskum)</b>	<b>48</b>
<b>4.1 Vzor otázok</b>	<b>49</b>
<b>4.2 Výsledky</b>	<b>49</b>
<b>5. Analýza hudby</b>	<b>50</b>
<b>6. Implementácia</b>	<b>61</b>
<b>6. 1 FFT</b>	<b>61</b>
<b>6. 2 Beat Detection</b>	<b>64</b>
<b>6. 3 Music_analysis</b>	<b>67</b>
<b>6. 4 Konfrontácia výsledkov</b>	<b>70</b>
<b>Záver</b>	<b>73</b>
<b>Zdroje</b>	<b>74</b>

# **Obsah obrázkov**

<b>2. Zvuková analýza</b>	
<b>2-1 Zvuková vlna</b>	<b>21</b>
<b>2-2 Kombinácia zvukových vín</b>	<b>22</b>
<b>2-3 Diskrétna fourierova transformácia</b>	<b>25</b>
<b>2-4 Motýlikový diagram</b>	<b>27</b>
<b>2-5 Cooley-Tukey (usporiadanie vstupov)</b>	<b>29</b>
<b>3. Beat detection</b>	
<b>3-1 Schéma BTS</b>	<b>37</b>
<b>3-2 Fázy systému BTS</b>	<b>40</b>
<b>3-3 Detekcia nástupu</b>	<b>41</b>
<b>3-4 Extrahovanie hluku (detekcia malého bubna)</b>	<b>42</b>
<b>3-5 Výber frekvencie basového bubna</b>	<b>43</b>
<b>3-6 Inter onset interval histogram</b>	<b>45</b>
<b>3-7 Forma preddefinovaných bubenových vzorov</b>	<b>45</b>
<b>3-8 Porovnávanie detekovaného beatu so vzorom</b>	<b>46</b>
<b>5. Analýza hudby</b>	
<b>5-1 Schéma systému na vnímanie hudby</b>	<b>52</b>
<b>5-2 Detekcia udalostí pomocou spektrogramu</b>	<b>55</b>
<b>5-3 Vypočítanie tempa zo spektrogramu</b>	<b>57</b>
<b>5-4 12 dimenzionálny vektor chroma</b>	<b>58</b>
<b>5-5 Chromogramy rôznych segmentov</b>	<b>58</b>
<b>5-6 Reprezentácia skladby (len klavír)</b>	<b>60</b>
<b>6. Implementácia</b>	
<b>6-1 Zvukové spektrum</b>	<b>63</b>
<b>6-2 Rôzne módy vizualizácie aplikácie FFT_Magic</b>	<b>63</b>
<b>6-3 Vizualizácia Beat detection algoritmu</b>	<b>66</b>
<b>6-4 Dáta získané pomocou triedy DuchEchoNest</b>	<b>69</b>

# Slovník pojmov

**tón** - základ hudby, zvuk, ktorý ma istú výšku (frekvenciu)

**interval** - vzdialenosť medzi tónmi

**stupnica** - postupnosť tónov rôznych intervalov

**tónina** - informácia, ktorá hovorí o názve stupnice, v ktorom je skladba skomponovaná

**akord** - súzvuk troch a viacerých tónov

**charakter skladby** - nadobúda možnosti dur alebo mol. Durová skladba má veselší charakter (obsahuje napríklad interval veľkej tercie - vzdialenosť dvoch tónov) a molová skladba smutnejší charakter (obsahuje napríklad malú terciu - vzdialenosť jeden a pol tónu).

**tempo** - rýchlosť skladby

**takt** - udáva počet daných nôt na jeden takt (napríklad 2/4 takt znamená, že jeden takt tvoria dve štvrtové noty)

**farba (timbre)** - každý nástroj vytvára tóny rôznej farby (nemyslí sa skutočná farba)

**mastering** - forma audio post-produkcie, v ktorej vzniká výsledný zvukový záznam, z ktorého sa vyrábajú všetky kópie.

# Zoznam skratiek

**FFT** - Fast Fourier Transform (rýchla fourierova transformácia)

**DFT** - Discrete Fourier Transform (diskrétna fourierova transformácia)

**DTMF** - Dual - Tone Multi Frequency signaling (telekomunikačné signály v telefónoch)

**BTS** - Beat Tracking System (systém na detekovanie beatov od Masataka Gotu)

**IBI** - Inter Beat Interval (interval medzi detekovanými beatmi)

**IOI** - Inter Onset Interval (interval medzi nástupmi)

**P2P** - Peer 2 peer

**MIT** - Massachusetts institute of technology

**MP3** - MPEG Audio Layer 3 kodek

**LLD** - Low level audio descriptor (opisuje zvuk na nízkej úrovni)

**BPM** - Beats per minute (úderov za minútu)

# Úvod

Najprv by som chcel objasniť, čo ma vlastne viedlo spracovať túto tému. Hudbe sa venujem od svojho útleho veku a téma jej vizualizácie mi bola vždy blízka, preto bolo získavanie informácií na danú tému logickým vyústením môjho voľného času. Funkciu vizualizérov v rôznych hudobných prehrávačoch som využíval celkom aktívne, najmä v mladom veku, keď doba počítačov nebola až tak graficky prítulná a preto každý plynulý, vizuálne atraktívny pohyb vyvolal v ľuďoch nadšenie. Čím som bol starší, tým sa mi vizualizéry hudby vzdalovali, aj keď vždy som ich vývoj aspoň sledoval. V mojom okolí ich ľudia veľmi nepoužívali a keď sa aj našli výnimky, ktoré používali vizualizér, jeho využitie bol skoro vždy vo forme podprahové doplnku pri upratovaní a iných neatraktívnych činnostach. Samozrejme, môžeme ešte spomenúť vizualizáciu ako doplnok pri rôznych hudobných výstupeniach, aj keď tieto vizualizácie často nemajú úlohu byť priamo vizualizáciou danou hudby, skôr doplniť umelecký zážitok o ďalší vnem .

Ako je to však priamo s vizualizáciou hudby v súčasnosti? Základom vzniku väčšiny vizualizácií je nasledovný postup. Vizualizér prečíta zo zvukového signálu dátu (väčšinou po veľmi krátkych úsekoch - menej ako 20ms) a na ne aplikuje metódy frekvenčnej analýzy. Jedna z najpoužívanejších takýchto metód je napríklad rýchla fourierova transformácia. Často sa k týmto dátam pridávajú údaje, ktoré informujú o beatoch a o ich nástupoch. Takto získame dostatočné množstvo variabilných premenných, ktoré užívateľ môže vizualizovať ako akékoľvek iné dátá.

Sú však takéto údaje dostatočné? Dá sa pomocou takýchto techník vytvoriť vizualizér, ktorý by naozaj zodpovedal hudbe, ktorá hrá? Má vôbec význam, aby vizualizér zodpovedal hudbe? Existujú ľudia, ktorí by takýto vizualizér vedeli využiť? Môže mať vizualizér aj informačnú hodnotu, alebo je jeho hlavná funkcia byť len vizuálne príťažlivý? Toto sú otázky, ktorým môžem ďakovať za vznik tejto práce.

# 1. História

Za prívou použiteľnú vizualizáciu hudby môžeme považovať noty. Keďže tento prístup vizualizácie neboli informatického charakteru, spomieniem ho len veľmi okrajovo.

Prvá zmienka o notách je z 5. storočia, keď rímsky filozof Boethius použil prvých pätnásť písmen z abecedy na označenie všetkých tónov (dve oktávy), ktoré boli v tej dobe používané. Aj keď nie je presné známe, či sa táto "vizualizácia" ujala, dodnes sa považuje za prívou známu notáciu pod názvom *Boethiusova notácia*.[3]

Prvá informaticky súvisiaca vizualizácia je asi produkt vyrobený Jeffom Minterom pod názvom *Psychedelia*. Bol publikovaný firmou Llamasoft v roku 1984. Samotný program nedostával na vstup žiadne informácie z audia, všetko ovládal užívateľ joystickom. Užívateľ mal na výber rôzne predefinované programy, alebo ak bol zdatnejší mohol si sám navoliť premenné, ktoré ovládali pulzy vysielajúce vďaka pohybu joysticku. Aj keď teda táto vizualizácia priamo nebrala informácie z audia, tak bola určené na doprevádzanie hudby.[4]

Za druhú informaticky akceptovateľnú vizualizáciu, ktorá spolupracovala už aj so samotným zvukom môžeme považovať produkt firmy *Infinite Software*, ktorý vydali v roku 1985 a predávali ju za sumu desať libier. Názov produktu bol **Sound to Light generator** a bol určený pre osem bitový počítač *ZX Spectrum* s pamäťou 48KB.[5] Snažil som sa zistiť na akom princípe fungovala táto vizualizácia, bohužiaľ nenašiel som dostatok informácií.

Ďalší významný produkt v tomto odvetví bol softvér **Cthugha** z roku 1993. Bola napísaná austrálskym študentom *Kevinom Burfittom* primárne pre platformu DOS. Neskôr však bola implementovaná aj pre Linux (1995) a Macintosh (1996). Vstupom tohto programu boli rôzne audio zdroje, ako napríklad mikrofón, CD mechanika (musela byť zapojená do zvukovej karty) alebo nejaký audio súbor.

*Cthugha* spracovala audio informácie, prehnala ich cez rôzne filtre a zobrazovala výsledok na obrazovke v reálnom čase. Užívateľ mohol meniť rôzne parametre zobrazovanie pomocou klávesnice, mohol si vybrať jeden určitý parameter alebo sa nechat unášať náhodným výberom. Tento program sa často nazýval “oscilloscope on acid”, pretože osciloskop bol naplnený audio informáciami a vznikali rôzne obrazce, ktoré pripomínali drogové psychadélie. Tento softvér ponúkal už dokonca aj možnosť použitia FFT (Fast Fourier Transform), ktorý pridával efektom viacdimenzionálnejší charakter. Užívateľ si mohol vyberať z veľkého množstva filtrov a následne ich aj vrstviť, čím sa často stávalo, že následná vizualizácia nemala s danou hudbou nič spoločné. Na tú dobu to bola však nevídaňaná, vizuálne príťažlivá zábavka, ktorú ľudia brali celkom vážne. Tým, že bola v podstate užívateľsky ovládateľná, ľudia sa v nej zdokonaľovali a vizualizácie boli často subjektívne. Dokonca som našiel jeden názor, že daný človek sa pri vytváraní vizualizácie cítil ako keby hral na hudobnom nástroji. Ľudia tomu venovali veľa času a tí pokročilejší vedeli vytvárať vkusné vizualizácie skoro pri každej pesničke (empiricky zistili aké filtre sa hodia na aký štýl hudby, s tým, že parametre boli čisto subjektívou záležitosťou). Kevin Burfitt zastavil vývoj *Cthughy* v roku 2001, boli aj pokusy znova oživiť tento projekt, avšak všetky boli neúspešné hlavne kvôli veľkému množstvu podobného softvéru. [6]

Následne sa podobné aplikácie množili veľmi rýchlym tempom a to hlavne kvôli takým softvérom ako WinAmp, Audion a SoundJam. Významné miesto na vizualizačnom trhu získalo *Advanced Visualization Studio* od Nullsoftu, *MilkDrop* od *Ryan Geissa* a *G-Force* od Andyho O'Meara. Hlavným rozdielom medzi týmito vizualizáciami a inými formami vizualizácie hudby (videoklipy, laserové šou) je ten, že vďaka veľkému počtu náhodných premenných vytvára rôzne vizualizácie aj pre rovnaké piesne. V súčasnosti sú najpoužívanejšie práve *MilkDrop*, ktorý je zdarma k prehrávaču WinAmp a G-Force, ktorý je súčasťou iTunes a Windows Media Player.

## 1. 1 MilkDrop

Tvorcom tohto pluginu je už vyššie spomínaný Ryan Geiss, prvá verzia vyšla v roku 2001. Od roku 2005 prevzal vývoj tohto softvéru Geoff Potter a pracuje naňom v podstate dodnes. MilkDrop je v podstate len prostredie, ktoré slúži na obsluhu a spúšťanie vopred definovaných “presetov”. Užívateľ má možnosť v reálnom čase prepínať tieto presety, pomocou vizuálne atraktívnych, zážitok nekaziacich prechodov. MilkDrop navyše pridáva možnosť vytvárať si vlastné nové presety, ktorých sa následne stávate autorom. Po internete je veľké množstvo fórum, stránok, na ktorých si užívatelia medzi sebou vymieňajú sebou vytvorené presety. Každý preset je uložený v .milk formáte, a obsahuje 4 hlavné skriptovateľské sekcie. Patria sem *per\_frame*, *per\_pixel*, *custom shapes* a *custom waves*. [7]

V sekcií *per\_frame* sa premenné aktualizajú, ako vyplýva z názvu, každý frame. Tieto nové hodnoty premenných sa potom posielajú do iných sekcií, kde sa modifikujú vlastnosti celkovej vizualizácie ako napríklad interakcie s audio informáciami (získané cez FFT). Táto sekcia riadi celkový prístup k tomu ako sa budú premenné a teda celá vizualizácia vyvíjať.

Ďalšia sekcia je *per\_pixel*. Z názvu by mohlo vyplývať, že je to kód, ktorý sa aktualizuje pre každý pixel, nie je to však pravdou. Obrazovku MilkDrop rozdeľuje do vopred danej mriežky a táto sekcia ovplyvňuje každý bod zadefinovanej mriežky. Pixle, ktoré sú medzi týmito bodmi získavajú svoje vlastnosti pomocou štyroch ich obklopujúcich bodov v mriežke. Vopred daná veľkosť mriežky je 32x24, môže byť však upravená užívateľom na menšiu, alebo väčšiu.

V sekciách *custom shapes* a *custom waves* vystupujú premenné, ktoré autorovi presetu dovoľujú meniť tvar, veľkosť, farbu, pozíciu a iné parametre. Aj tvary aj vlny majú vlastnú *per\_frame* sekciu, ktorá ovplyvňuje ich premenné počas každého frameu. Primárne bolo povolených na jeden preset štyri tvary a štyri vlny, v poslednej beta verzii je ich povolený počet už päť.[8]

MilkDrop môže používať rôzne audio zdroje (cd, mikrofón, line in), čím sa stáva použiteľným aj v reálnom čase a preto je často oblúbeným nástrojom VJov. Posledná verzia MilkDrop-u bola vydaná v roku 2007 a podporovala pixel shadre. Primárnym využitím tohto pluginu je program WinAmp a platforma Windows, avšak po veľkom úspechu vznikli rôzne projekty, ktoré sa snažili implementovať MilkDrop aj do iných softvérów. Keďže MilkDrop používa DirectX, ktorá je výslovne "windowsovou" záležitosťou, bolo treba prepísať ho do OpenGL. Toto dokázal projekt *projectM*, ktorý bol vydaný ako plugin do softvérów ako iTunes, Audacious, XMMS, Jack, PulseAudio a FooBar2000. MilkDrop bol dokonca portovaný aj do Xbox Media Center platformy, ktorá sa používa v Xboxoch a na PC (to však bola oficiálna podpora).[7]

## 1. 2 G - Force

Aplikáciu *G - Force* vyvíja firma *SoundSpectrum*, predtým známa ako *WhiteCap Technologies*. Jej autor je Andy O'Meara a nebol to jeho prvý vizualizačný projekt. Ten prvý bol vydaný v roku 1999 pod názvom WhiteCap. Nasledovníkom bol projekt *G - Force* vydaný o rok neskôr. Dodnes obidva produkty fungujú pod záštitou Andyho firmy SoundSpectrum a dodnes sa z nich stiahlo desiatky miliónov kópií.[9]

*G - Force* sa stal nesmierne populárny najmä kvôli možnosti využiť vizualizácie rôznymi spôsobmi, ako šporič obrazovky, samostatnú animáciu, dokonca sa dali zábery z nej aj tlačiť. Každé 2 mesiace vydáva táto spoločnosť platené aktualizácie, kde dodáva užívateľom nové dizajny a možnosti pre ich vizualizácie .

V roku 2001 dostal ponuku od formy Apple aby sa *G-Force* stal hlavným vizualizačným prostriedkom ich nového multimediálneho prehliadača *iTunes*. Narozdiel od iných voľne dostupných vizualizačných prostriedkov poskytuje rôzne verzie aplikácie. Najzákladnejšia, ktorá je zadarmo, obsahuje len základ a chýbaju jej funkcie ako toolbar na kustomizáciu a lepšiu kontrolu, rôzne kombinácie vizuálnych efektov, možnosť práce s názvami pesničiek (z id tagov), obalmi

albumov a iné. Všetky tieto pridané možnosti sú dostupné až v Gold edícií, ktorá stojí dvadsať dolárov. Pre ozajstných nadšencov je dostupná aj Platinum edícia, ktorá obsahuje podporu pre väčšinu prehrávačov a taktiež možnosť využívať vizualizáciu ako šporič obrazovky, samostatnú aplikáciu (využiteľná pri vizualizácii audia z rôznych zdrojov ako aux, mikrofón, internetové rádia a pod.). Taktiež dostane zákazník pri zakúpení Platinum edície, ktorá stojí 30 dolárov, zadarmo všetky aktualizácie, ktoré vyjdú v období jedného roka od zakúpenia.[10]

Čo sa týka technologického zázemia, narození od MilkDrop-u nemá G-Force zverejnený zdrojový kód, takže je pomerne ťažké zistiť aké technológie používa. Autori uvádzajú len základné informácie, že využívajú kombináciu beat - detection algoritmov a dát získaných pomocou rôznych techník analýzy frekvencie

## 2. Zvuková analýza

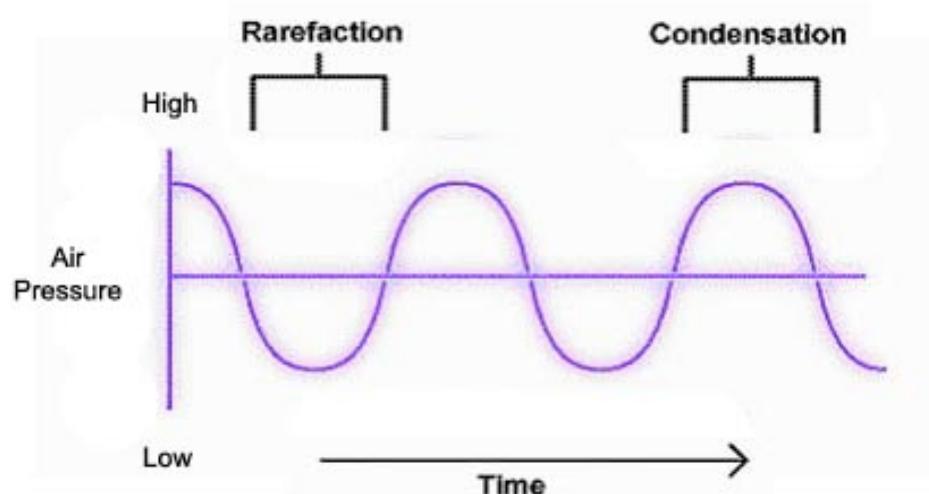
V tejto kapitole budeme riešiť rôzne prístupy ako analyzovať audio signál. Nato aby sme to boli schopní zvládnuť, budeme si musieť vysvetliť, čo to vlastne audio signál alebo frekvencia je, ako ju človek vníma, a ako ju môžme ďalej spracovať na priateľnejšie údaje. Analýza zvukových signálov však nie je dôležitá len kvôli našej téme, čo je vizualizácia zvuku. Pomocou analýzy zvuku vieme získať množstvo informácií, ktoré sa využívajú v rôznych oblastiach, najmä však v hudobnom priemysle. Pri nahrávaní hudobných nosičov je jedna z posledných a veľmi dôležitých činností práve *mastering*, ktorý ma na starosti práve analýzu audia a upravovanie hlastostí rôznych frekvencií tak, aby bol zvuk vyrovnaný. Človek sa nemôže spoliehať len na svoje uši, pretože zvuk, ktorý počuje môže byť skreslený rôznymi faktormi (zvukové monitory, miestnosť atď), a preto môže znieť v iných priestoroch úplne inak, taktiež uši nevedia rozoznať takzvané frekvenčné maskovanie (ak majú dva zvuky podobné frekvencie, môže sa stať, že jeden pohltí a schová ten druhý). A práve v týchto problémoch pomôže zvuková analýza, ktorá ukáže pravú tvar zvuku.

### 2. 1 Úvod o zvuku

Najjednoduchšie povedané, zvuk je vibrácia. Zvuk dokáže prechádzať cez rôzne substancie ako sú tekutiny, plyny alebo pevné objekty. Dokonca tieto substancie, alebo vodiče zvuku, ako sa im tiež hovorí, potrebuje, pretože napríklad vo vákuu sa zvuk nešíri. Pre náš druh zvuk presnejšie znamená vibrácie zložené z frekvencií, ktoré je schopné náš sluchový aparát (icho) počuť. Aby sme boli presní, naše ucho dokáže počuť frekvencie od 20 Hz až po 20 kHz (20 000 Hz), samozrejme vekom sa hlavne horná hranica neustále znižuje.[11]

Aby sme boli presnejší v našej definícii, musíme si uvedomiť, že vyššie spomínane vibrácie prechádzajú cez vodiče v podobe zvukových vln. Ako vlny prechádzajú cez materiál, molekuly do seba narážajú a vracajú sa na svoju

pôvodnú pozíciu. Výsledkom toho je, že oblasti vodiča sa stávajú hustejšími (condensation) alebo menej hustými (rarefaction).



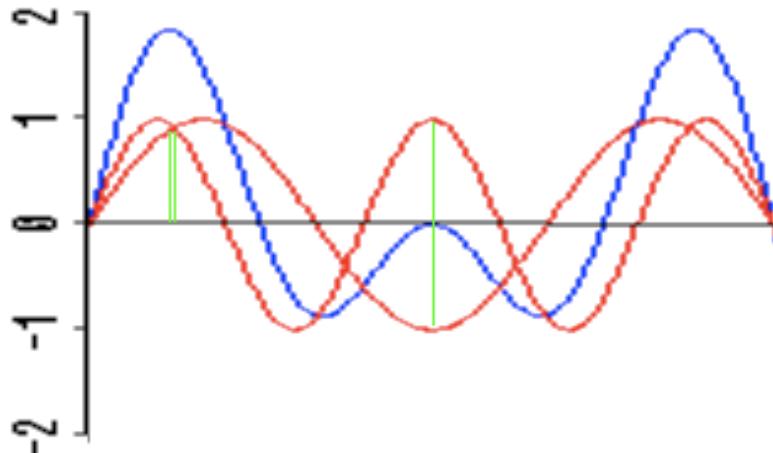
obrázok 2-1 Zvuková vlna [11]

Na obrázku 2-1 vidíte zvukovú vlnu v jej najčastejšej reprezentácii. A to pomocou grafu, kde x ová súradnica je už typicky čas a y ová súradnica reprezentuje tlak alebo hustotu vodiča, cez ktorý zvuk prechádza[13].

Každá zvuková vlna ma štyri základné vlastnosti a to sú - vlnová dĺžka, períoda, frekvencia a amplitúda. Vlnová dĺžka je horizontálna dĺžka jedného cyklu zvukovej vlny (vzdialenosť medzi dvoma po sebe idúcich ekvivalentnými bodmi). Períoda je čas potrebný na absolvovanie jednej vlnovej dĺžky. Amplitúda je daná výškou danej zvukovej vlny (presnejšie maximálnou hodnotou na ypsilonovej osi), čím je výška väčšia, tým je aj samotný zvuk hlasnejší. Ako sme si už povedali, každý cyklus sa skladá zo sekcie, kde je tlak vzduchu (alebo iného vodiča) väčší (condensation) a zo sekcie, kde je menší (rarefaction). Frekvencia hovorí o počte cyklov za sekundu. To znamená, že ak ma zvuk frekvenciu 20 000 Hz, za sekundu prebehne 20 000 zhustení a zriedení. Čím ma zvuk väčšiu frekvenciu ( viac zhustení a zriedení za sekundu) tým je tón vyšší.[14]

Zvukové vlny sa dajú kombinovať a týmito kombináciami vznikajú nové vlny. Ak teda naraz počujete dve vlny z dvoch zdrojov platia nasledujúce pravidlá. Vysoký tlak zruší jeho presný opak, dva vysoké tlaky sa spočítajú, dva nízke sa

tiež spočítajú. Najlepšie to vidíte na obrázku 2-2. Dve červené vlny vytvorili modrú vlnu. Pri prvej zelenej čiare vidíte ako sa spočítali hustoty a pri druhej zelenej čiare zasa máme možnosť vidieť ako sa navzájom vynulovali. [12]



obrázok 2-2 Kombinácie zvukových vín, [12]

Teraz ked' už vieme, ako vyzerá zvuk v jeho naturálnej podobe, môžme si povedať základné informácie o tom, ako sa zvuk dá zdigitalizovať. Digitalizácia zvuku používa na reprodukciu zvuku digitálne signály. Digitálny zvuk má dve hlavné premenné: sample rate a bit resolution. Sample rate hovorí o tom, koľko samplov potrebujeme za sekundu na prevod zo spojitého signálu na diskrétny. Bit resolution hovorí o tom, koľko bitov informácií je nahratý pre každý sample. Zvyčajná sample rate je 44 100 samplov za sekundu a napríklad CD kvalita ma 16bitové rozlíšenie.

Tieto základné vedomosti nám umožnia lepšie pochopiť ďalšie sekcie, kde už budeme spomínať konkrétné spôsoby analýzy zvuku a získania samotných informácií z jeho vlastností.

## 2. 2 Fourierova transformácia

Ako sme si vyššie povedali zvuk je reprezentovaný zvukovou vlnou, to znamená že typický signál je reprezentovaný ako sínusoidná krvka, ktorá stúpa a klesá počas istého časového intervalu. Avšak to nie je úplne presne princíp na ktorom funguje nás mozog. Mozog sa sústredí viac na frekvencie a ich hlasitosti ako na hustoty alebo tlak média, cez ktoré daný zvuk prechádza. Prirodzene sa teda snažíme vizualizovať zvukový signál ako graf, ktorý má na x ovej osi frekvencie a na y ovej osi hlasitosť. Na dekompozíciu signálu na jeho frekvencie poznáme takzvanú fourierovu transformáciu. Bohužiaľ, jeho najjednoduchšia implementácia je tak zaťažujúca a náročná, že je nereálne, aby bežala v reálnom čase. Naďastie existuje modifikácie tejto transformácie, známa ako FFT (fast fourier transform), ktorá zvláda túto konverziu v reálnom čase. A v tejto sekcií si povieme bližšie niečo o fourierových transformáciach a jej modifikáciach.

Takže čo je to vlastne Fourierova transformácia? Podľa [18] slúži hlavne na prevod obrazu do dualného priestoru, ktorý zjednodušuje niektoré operácie s obrazom a je vhodný pre lepšie pochopenie niektorých javov a vlastností obrazu. Fourierov obraz je v podstate reprezentácia obrazu v frekvenčnej oblasti. Frekvenčná oblasť je v reprezentácii, ktorá vzniká zložením nekonečného množstva sínusoidových signálov s rôznou amplitúdou a s rôznym fázovým posunom. Analógiou je reprezentácia zvuku ako signálu, ktorý sa skladá zo signálov nízkej a vysokej frekvencie. Čím hlasnejšie sú basy, tým je amplitúda nízkych frekvencií väčšia a naopak, čím hlasnejšie sú výšky, tým väčšia je amplitúda vysokých frekvencií. Samozrejme, občas sa stane, že niektoré frekvencie sú nežiadúce, takéto frekvencie nazývame aliasy (ak majú nízku frekvenciu) a šumy (vysoká frekvencia ).

Samotný výpočet, ktorým získame Fourierov obraz z funkcie reprezentovanej v časovej oblasti sa nazýva priama fourierova transformácia, Fourierov obraz funkcie  $f(x)$  tu definujeme ako komplexnú funkciu  $F(u)$ , ktorú získame integráciou

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx$$

kde  $i$  je komplexná jednotka ( $i = \sqrt{-1}$ ).

Inverzná fourierová transformácia komplexnej funkcie  $F(u)$ , ktorá realizuje prechod do priestorovej oblasti (z frekvenčnej) je definovaná vztahom

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{i2\pi ux} dx$$

Ked'že  $f(x)$  a  $F(u)$  sú vyjadrením tej istej funkcie spojenej reláciou Fourierovej transformácie, často sa ich vztahy označujú ako

$$f(x) \Leftrightarrow F(u)$$

## 2. 2. 1 Diskrétna Fourierova transformácia

Pri práci so signálom, ktorý je daný obmedzenou postupnosťou vzorkov, ktorá sa opakuje (to je presne prípad našej zvukovej témy) sa používa takzvaná diskrétna Fourierova transformácia (DFT). Priama a inverzná DFT funkcie  $I_k$  sú definované ako nasledujúce rovnice [18]

$$F_n = \sum_{k=0}^{N-1} I_k e^{-i2\pi \frac{nk}{N}}$$

$$I_k = \sum_{n=0}^{N-1} F_n e^{+i2\pi \frac{nk}{N}}$$

Dualitu medzi priestorovou  $I_k$  a frekvenčnou  $F_n$  vyjadríme pomocou zrejmého

$$I_k \Leftrightarrow F_n$$

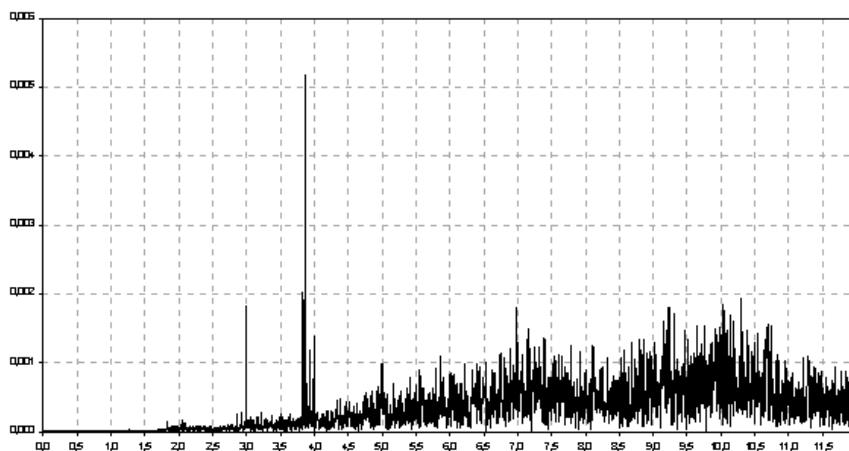
Fourierova transformácia je teda dekompozícia pôvodnej funkcie  $I_k$  na rôzne fázovo posunuté sínusoidné funkcie s rôznou amplitúdou. Teraz zadefinujeme dve dôležité premenné, ktoré charakterizujú obraz. Je daný bod komplexného Fourierovho obrazu so súradnicou  $u = \text{Re}(u) + i \text{Im}(u)$ . Jeho reálnu časť označujme  $\text{Re}(u)$  a imaginárnu časť  $\text{Im}(u)$ . Zadefinujeme *amplitúdové spektrum*  $|F(u)|$

$$|F(u)| = \sqrt{\text{Re}^2(u) + \text{Im}^2(u)}$$

a *fázové spektrum*

$$\varphi(u) = \arctan\left(\frac{\text{Re}(u)}{\text{Im}(u)}\right)$$

Amplitúdové spektrum slúži na určenie veľkosti jednotlivých frekvencií a fázové spektrum určuje ich posuny. Takže bod v samotnej Fourierovej oblasti, ktorý ma frekvenciu  $u$  Hz, má amplitúdu  $|F(u)|$  a fázový posun  $\varphi(u)$ .



obrázok 2-3 Diskrétna Fourierova transformácia - amplitúdové spektrum , [15]

## 2. 2. 1. 1 Rekurzívny rozklad DFT

V roku 1942 s týmto rozkladom prišli na svet Danielson a Lanczos. Dokázali, že DFT dĺžky N môže byť nahradená súčtom dvoch DFT s dĺžkou N/2, kde N je celé číslo, ktoré je mocninou dvojky. Z pôvodnej postupnosti N bodov transformuje jedna DFT vzorky s párnym indexom a druhá s nepárnym indexom. Na základe týchto informácií postupne upravíme základný vzorec pre DFT. [19]

$$F_n = \sum_{k=0}^{N-1} I_k e^{-i2\pi \frac{nk}{N}}$$

$$F_n = \sum_{k=0}^{(N/2)-1} I_{2k} e^{-i2\pi \frac{n(2k)}{N}} + \sum_{k=0}^{(N/2)-1} I_{2k+1} e^{-i2\pi \frac{n(2k+1)}{N}}$$

$$F_n = \sum_{k=0}^{(N/2)-1} I_{2k} e^{-i2\pi \frac{n(2k)}{(N/2)}} + W^n \sum_{k=0}^{(N/2)-1} I_{2k+1} e^{-i2\pi \frac{n(2k+1)}{(N/2)}}$$

$$F_n = F_n^e + W^n F_n^0$$

$F_n^e$  označuje n-tý člen DFT dĺžky N/2 postupnosti zloženej z párnych členov a  $F_n^0$  zasa označuje n-tý člen DFT tej istej dĺžky nepárnych členov. Z definície vyplýva, že  $F_{n+N} = F_n$ , to znamená že  $F_n$  je tiež periodická s periódou N.

Z toho teda tiež vieme povedať, že

$$F_{n+N/2}^e = F_n^e$$

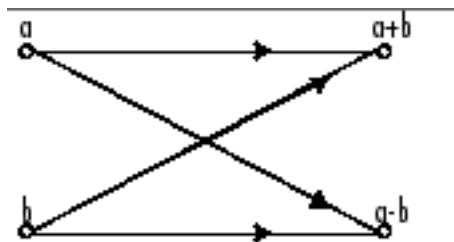
$$F_{n+N/2}^0 = F_n^0$$

Po dosadení dostaneme takzvané "motýlkové" usporiadanie výpočtu vyjadrené rovnicami.

$$F_n = F_n^e + W^n F_n^0$$

$$F_{n+N/2} = F_n^e - W^n F_n^0$$

Každý motýlik zoberie dve komplexné čísla (na obrázku 2-4) a vypočíta z nich ďalšie dve ako  $a + \alpha b$  a  $a - \alpha b$  kde  $\alpha$  je komplexné číslo.



obrázok 2 - 4 Motýlikový diagram [17]

## 2. 2. 2 Fast Fourier Transform

Tento algoritmus bol navrhnutý na základe vyššie uvedeného rozkladu a dokáže vypočítať DFT s použitím  $O(N \log N)$  násobení a sčítaní. Samotný algoritmus má však veľmi veľa variant ako napríklad Bruun's, Bluestein's, Rader's FFT algoritmy. My si ukážeme priamu realizáciu postupu rekurzívneho rozkladu a najpoužívanejší Cooley Tukey algoritmus. [19]

### 2. 2. 1 Rekurzívny FFT algoritmus

Rekurzívny FFT algoritmus, ktorý priamo realizuje postup rozkladu DFT si môžete pozrieť v tejto sekcií. Pre praktické implementácie sa však viac hodia nerekurzívne algoritmy ako Cooley-Tukey

---

```

procedure fft_recursive(N, f) {
    if N = 2 {
        h = a;
        a = a+b;
    }
}

```

```

        b = h - b;
    } else
        f1 = zoznam vsetkych bodov z f s parnym indexom
        f2 = zoznam vsetkych bodov z f s neparnym indexom
        FFT(N/2, f1);
        FFT(N/2, f2);
    }
=====
```

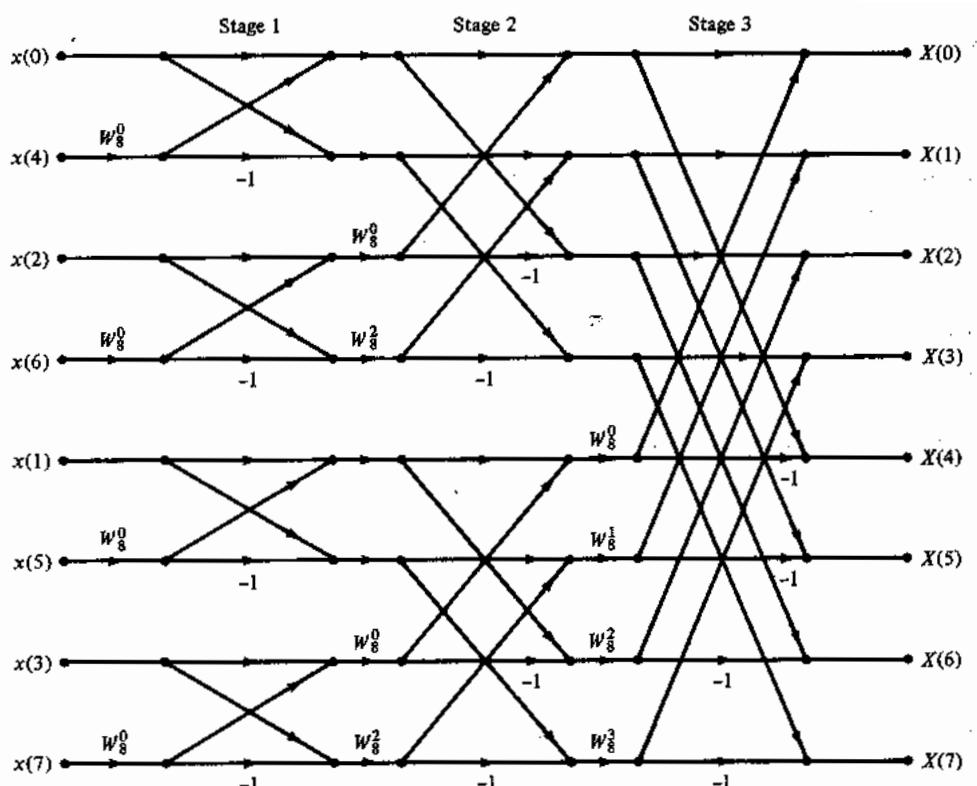
### **2. 2. 1. 2 Cooley - Tukey FFT algoritmus**

Tento algoritmus je teda nerekurzívny a tiež sa nazýva *decimácia v čase*. Na obrázku nižšie vidíte obrázok (motýlikový diagram) osem bodovej FFT. Výpočet tohto algoritmu rozkladá DFT na  $\log_2 N$  sekcií, každá sekcia je zložená z  $N/2$  motýlikových operácií.

Ako ste si všimli, na vstupe vstupujú vzorky po dvojiciach v takomto poradí -  $(x(0),x(4)), (x(2),x(6)), (x(1),x(5)), (x(3),x(7))$ . Táto zmena vyzerá na prvý pohľad zložito, indexy však v sebe skrývajú veľmi jednoduché tajomstvo. Všetko nám obzrejmí nasledovná tabuľka.

pozícia na vstupe (p1)	0	4	2	6	1	5	3	7
pozícia na výstupe (p2)	0	1	2	3	4	5	6	7
p1 (binárne)	000	100	010	110	001	101	011	111
p2 (binárne)	000	001	010	011	100	101	110	111

Ako vidíte, ak zapíšeme premenné p1 a p2 v binárnom zápise, zistíme, že každé p1 je bitovo prevrátené p2. Algoritmus Cooley-Tukey teda pracuje tak, že pred samotným výpočtom preusporiada vstup, podľa požadovaného výstupu, ktorý len bitovo prevráti. Pri dokončení výpočtu tak používateľ vlastne dostane výstup už v prirodzenom poradí (ako vidíte na obrázku 2-5).



obrázok 2-5 Cooley-Tukey FFT algoritmus (usporiadanie vstupov) [16]

## 2. 2. 3 Goertzelov algoritmus

Goertzelov algoritmus je primárny algoritmus na detekciu tónov. Samozrejme, na takúto úlohu môžeme použiť aj FFT, avšak pri detekovaní len párov tónov zo zvukového signálu to môže byť dosť nepraktické, keďže FFT je oveľa náročnejšia na CPU ako tento algoritmus. Takže aj keď často tento algoritmus uniká pozornosti, ak máme za úlohu detektovať len jeden alebo viac tónov (frekvencií), tento algoritmus je momentálne určite jeden z najrýchlejších dostupných.

Tento algoritmus bol pomenovaný podľa jeho autora Dr. Geralda Goertzela a bol publikovaný v roku 1958 [25]. Ako som spomenul vyššie, je to primárny algoritmus pre detekciu tónov a využíva sa napríklad v technológií DTMF (dual-tone multi-frequency signaling), ktorý sa používa v telefónoch, taktiež sa tento algoritmus používa priamo na dekodovanie priebehu hovoru (obsadené, vyzváňacie tón a podobne).

A kedy sa vlastne oplatí použiť Goertzelov algoritmus miesto FFT? Empiricky dokázali, že vtedy ak potrebujeme menej ako  $2\log_2 N$  DFT koeficientov ( diskrétna fourierova transformácia s N bodmi), napríklad v DTMF nám stačí prvých 8 tónov z celkového počtu, napríklad 205 DFT koeficientov. V takom prípade je jasné, že je zbytočné, aby sme rátali zvyšných 197 koeficientov. Takže hlavný rozdiel je, že kým FFT sa musí pozerať na celé spektrum zvukového signálu, Goertzelov algoritmus porovnáva len s vopred zadanými bodmi. [24]

# 3. Beat detection

Ako to už väčšinou býva, to čo je pre človeka najprirodzenejšie, býva najkomplikovanejšie prepísať do informatického jazyka. Nie je to inak ani pri téme tejto podkapitoly. Detekciu beatu alebo rytmu spracováva ľudský mozog plne automaticky a tak, že si to ani neuvedomujeme. Nikto z nás nikdy nerozmýšľal ako je možné, že si začneme zrazu podupkávať do rytmu a že vôbec rozoznáme, kde ten rytmus je, dokonca ho vieme trochu predvídať (ak nejde o isté free jazzové nahrávky). Pri prepisovaní do digitálneho sveta, to také jednoduché nebude, pretože počítač nepozná procedúru *feel\_rhythm* a v podstate dokáže pracovať len s logickými operáciami.

## 3. 1 Simple beat detection

Ako to vlastne funguje v našom mozgovom aparáte? Zvukový signál príjmaný ľudským sluchom sa skladá z istej energie, táto energia je konvertovaná do elektrického impulzu, ktorý je interpretovaný mozgom. Beat nastáva podľa tejto definície vtedy ak nastáva v tejto energie veľká zmena oproti predchádzajúcim energiam. Môžme si to predstaviť napríklad tak, že ucho príjma jeden monotónny signál, ktorý je občasne prerušovaný inými silnejšími signálmi, silnejšie signály pochopíme ako beat. Ak pustíme jeden hlasný signál bezozmien, žiadnen beat nerozoznáme. Zjednodušene teda môžme beat charakterizovať ako zmenu energie.

Podľa [26] je najjednoduchší prístup taký, že vyrátame priemernú energiu, ktorú potom porovnávame s energiou v danom momente. Predstavme si, že máme dve premenné  $a(n)$  a  $b(n)$ . V týchto premenných sú zoznamy amplitúd zaznamenaných každých  $T$  sekúnd pre ľavý a pravý kanál (stereo mód). Z týchto premenných potrebujeme vyrátať energiu v danom momente. Všeobecne považujeme za inštantnú energiu energiu, ktorá sa nachádza v 1024 vzorkoch (1024 hodnot  $a[n]$  a  $b[n]$ ), 1024 vzorkov (samplov) je v podstate asi päť stotín sekundy, takže sa to dá v kľúde považovať za inštantnú energiu.

Ked' už vieme vyrátať energiu v danom momente, potrebujeme ešte priemernú hodnotu. Samozrejme, nemôžeme brať priemernú hodnotu z celého hudobného diela, keďže skladba môže mať dynamicky slabšie a silnejšie časti, vždy musíme porovnávať s energiou, ktorá je blízka k inštantnej na časovej osi. Napríklad ak má skladba hlučný záver, nesmie to ovplyvniť tichý začiatok. Inak povedané náš algoritmus zachytí "beat" len vtedy ak je inštantná energia vyššia ako lokálna priemerná energia.

Vo všeobecnosti sa ráta priemerná energia na 44100 vzoriek (samplov), čo predstavuje asi 1 sekundu, čím vlastne určujeme, že náš program si bude pamätať jednu sekundu pesničky a v nej bude detektovať "beat". Náš history buffer kde ukladáme 44100 samplov bude v podstate obsahovať dva zoznamy samplov (B[0]) a (B[1]), ktoré korespondujú dvom stereo kanálom.

### **jednoduchý beat detection algoritmus //každých 1024 samplov**

**1.** zoberiem 1024 samplov z a[n] a b[n] a vypočítam inštantnú energiu

$$e = e_{\text{stereo}} = e_{\text{pravy}} + e_{\text{levy}} = \sum_{k=i_0}^{i_0+1024} a[k]^2 + b[k]^2$$

i0 je pozícia odkiaľ sa má brať 1024 samplov

**2.** vypočítam lokálnu priemernú energiu E na 44100 vzorkoch history buffera

$$E = \frac{1024}{44100} \times \sum_{i=0}^{44100} (B[0][i])^2 + (B[1][i])^2$$

B[0] a B[1] sú zoznamy samplov pre ľavý a pravý kanál

**3.** posuniem 44100 samplov veľký history buffer o 1024 indexov doprava, čím si spravím miesto pre nové sample a najstarších 1024 samplov zmažem.

**4.** najnovších 1024 samplov dám na vrch history buffera.

**5.** porovnám e a ( $c^* E$ ), pričom c je nejaká konštanta, ktorá určuje vlastne citlivosť detekovania beatu. Dobrá a veľmi používaná hodnota je 1.3. To znamená,

že ak je práve zachytená energia 1.3 násobkom priemernej energie (lokálnej) zachytili sme beat.

Toto je základná verzia algoritmu a veľmi jednoducho môžme vylepšiť jeho rýchlosť a presnosť. Jedna z hlavných optimalizácií môže byť tá, že miesto toho, aby sme si ukladali v histórii všetkých 1024 samplov, budeme si ukladať len hodnoty energií namerané na daných samploch. To bude znamenať, že pri rátaní priemerných energiách na 44100 samploch, nebudem pracovať so všetkými samplami ale len z hodnotami vyrátaných energií (nepoužijeme history buffer  $B$  ale nejaký iný history buffer, kde budú uložené už vyrátané instantné energie, nazvime si ho  $En$ ). Tento náš nový buffer  $En$  by mal obsahovať približne jednu sekundu nášho muzikálneho zdroja, takže v podstate obsahovať história energií cez 44100 samplov (skupiny po 1024 vzorkách - instantné energie). Takže  $En[0]$  bude obsahovať najnovšiu instantnú energiu vyrátanú na posledných, najnovších 1024 samploch a  $En[42]$  zasa energiu vyrátanú na najstarších 1024 samploch. V tomto našom novom buffri teda bude 43 hodnôt, pomocou ktorých budeme mať prehľad o energiách nameraných počas 1 sekundy.

### **optimalizovaný beat detection algoritmus //každých 1024 samplov**

**1.** znova vyrátame inštantnú energiu  $e$  na nových samploch

$$e = e_{\text{stereo}} = e_{\text{pravy}} + e_{\text{lavy}} = \sum_{k=i_0}^{i_0+1024} a[k]^2 + b[k]^2$$

$i_0$  je pozícia odkiaľ sa má brať 1024 samplov

**2.** Vyrátame priemernú lokálnu energiu  $E$  pomocou nášho nového history buffera  $En$

$$E = \frac{1}{43} \times \sum_{i=0}^{43} (En[i]^2)$$

3. Posunieme buffer  $En$  o jeden index doprava. Spravíme tým miesto pre novú hodnotu instantnej energie a najstaršiu zmažeme.
4. Na miesto  $En[0]$  dáme najnovšiu energiu  $e$ .
5. Porovnáme  $e$  s  $(C \times E)$ .

Ako som už spomínať vyššie, veľkú rolu v behu tohto algoritmu hrá konštanta C. V niektorých hudobných štýloch (techno, rap) sú beaty veľmi intenzívne, takže by mala byť konštanta dostatočne veľká, aby náhodou nezachytávala rôzne nechcené zmeny energií (napríklad 1.4), ale naopak napríklad v rockovej a hardrockovej hudbe, ktorá je sama o sebe dosť hlasná, musíme voliť menšiu konštantu (okolo 1.1). Ak by sme to vždy zadávať konštantu manuálne, nebola by to veľká výhra a preto nás jednoduchý algoritmus ešte zo optimalizujeme tak, aby vedel sám zistiť výhodnú voľbu pre našu konštantu.

Základným princípom je vyrátať priemer zmien energií, ktoré obsahuje nás "nový" buffer  $En$ , ktorý sme si pridali v prvej optimalizácii. Zmenu jednej energie vyrátame tak, že od inštantnej energie odrátame priemernú - to znamená  $(En - E)$ , presnejšie  $(En[0] - E)$ . Týmto dosiahneme rozdiel medzi energiou  $En[0]$  a  $E$ , ďalej vyrátame všetky rozdiely z 43 hodnôt, ktoré má nás buffer  $En$  a z nich nakoniec urobíme priemer. Čím väčší tento priemer bude, tým väčšie zmeny sa dejú a tým pádom potrebujeme menšiu konštantu C. Vo väčšine prípadov sa určujú vzťahy týchto premených takto, ak  $P(\text{priemer}) \geq 200$  tak  $C = 1.00$ , a ak  $P \leq 25$  tak  $C = 1.45$ . Na základe týchto predpokladov vyrátame konštantu C. Takto teda nakoniec vyzerá ďalšia optimalizácia tohto základného beatdetection algoritmu.

### optimalizovaný beat detection algoritmus #2 //každých 1024 samplov

1. znova vyrátame inštantnú energiu  $e$  na nových samploch

$$e = e_{\text{stereo}} = e_{\text{pravy}} + e_{\text{levy}} = \sum_{k=i_0}^{i_0+1024} a[k]^2 + b[k]^2$$

$i_0$  je pozícia odkiaľ sa má brať 1024 samplov

2. Vyrátame priemernú lokálnu energiu  $E$  pomocou nášho nového history buffera  $En$

$$E = \frac{1}{43} \times \sum_{i=0}^{43} (En[i]^2)$$

**3.** Vyrátame priemer zo zmien energií v  $En$

$$P = \frac{1}{43} \times \sum_{i=0}^{43} (En[i] - E)^2$$

**4.** Vyrátame konštantu C

$$C = -(0.00251714) \times P + 1.5142857$$

5. Posunieme buffer  $En$  o jeden index doprava. Spravíme tým miesto pre novú hodnotu instantnej energie a najstaršiu zmažeme.
6. Na miesto  $En[0]$  dáme najnovšiu energiu  $e$ .
7. Porovnáme  $e$  s  $(C \times E)$ , ak  $e > (C \times E)$  máme beat.

Tento algoritmus na beat detection je najjednoduchší, je veľmi presný a účinný, hlavne pre muziku techno a rap, kde sú beaty veľmi presné a hudba obsahuje málo ruchov (myslíme tým ruchov na energicky podobnej úrovni ako je beat). Avšak pri použití na punkovej, rockovej a inej hlasnej hudbe to nie je dostačujúci výsledok, napríklad nebude detektovať niektoré významné beaty, kvôli tomu, že budú utopené v skreslených gitarách.

Tiež napríklad majme dva nástroje, husle a flautu, ktoré hrajú na preskačku, to znamená, že keď jeden skončí druhý začne. Hrajú rovnakou hlasitosťou ale iné tóny, náš mozog zaregistrouje istý rytmus, pretože nástroje hrajú rôzne vysoké tóny, ale náš program nezaregistrouje nič, pretože energie sa nemenia. Takže tu je ďalší prípad, keď tento algoritmus nebude veľmi účinný.

Ako sa však naňho pozeráte, určite vidíte, že keďže je veľmi jednoduchý, je aj výpočtovo nenáročný, takže ak ľudia nehľadajú dokonalú beat detekciu a chcú skôr zvýrazniť veľký výkyv v energiách, prípadne v hudbe, kde nezná veľa nástrojov cez seba, je to tá pravá voľba.

## 3. 2 BTS

Tento systém BTS (beat tracking system), ktorý vytvoril Masataka Goto v roku 1995 [20] bol jeden z prvých systémov, ktoré dokázali extrahovať informácie o beate priamo z akustických signálov a nie z rôznych iných zdrojov ako napríklad MIDI (model Rosenthala a Browna z roku 1993). Podrobnejšie rozoberiem tento systém v nasledujúcej kapitole, zjednodušene sa dá povedať, že tento systém vyextrahoval z akustického signálu bicie a v nich hľadal a získaval beat. Prvý základný systém neriešil problém, ak by sa v signále bubny nenachádzali, neskôr výskum dokázal vyriešiť aj tieto problémy. Podme sa teda pozrieť na tento systém bližšie.

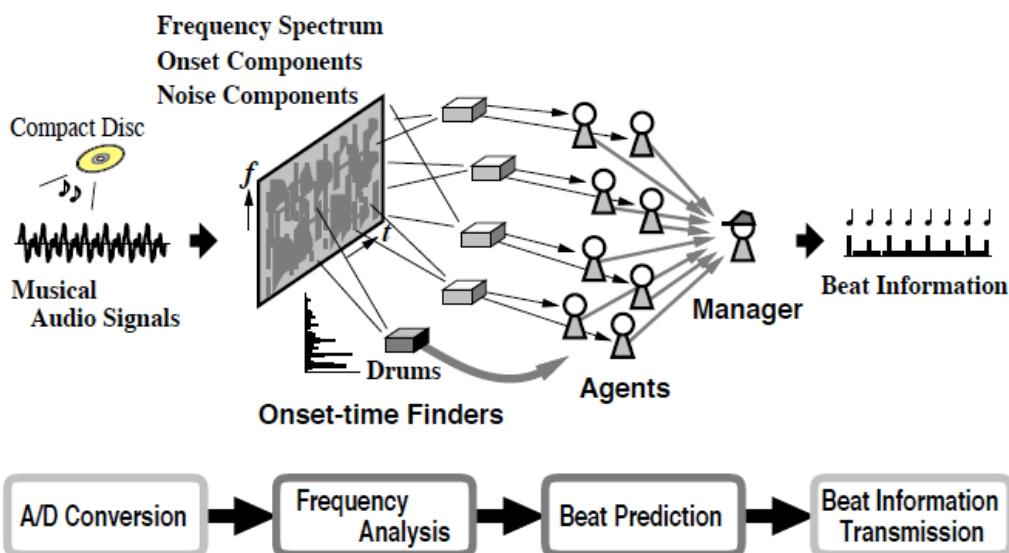
Tento systém spracovával audio signál a spoznával a určoval pozície beatov v reálnom čase. Signál obsahoval rôzne množstvo nástrojov a testovali ho najmä na populárnej hudbe ako pop, rock, a rôzne iné žánre, kde zabezpečujú hlavný rytmus bicie. Tento systém vedel predpovedať, kde a kedy asi príde nasledujúci beat (vzhľadom na šesťnástkovú notu, tých je v obyčajnom štvor-štvrtovom takte 16, takže celkom presné) a tiež vedel zistiť, či bol beat silný alebo slabý. Tieto pojmy znamenajú, či sa nachádza na výrazných dobách (prvá a tretia) alebo nevýrazných (druhá a štvrtá). Tieto pravidlá o výrazných a nevýrazných dobách sú len výsledkom zaužívaných hudobných zvykov, (prvá doba väčšinou s akcentom doprevádzaná basovým bubenom, tretia doba zase s malým bubenom - rytmičákem).

Často sa môže stať, že pri analyzovaní audio signálu môžu vzniknúť nejasné, dvojzmyselné situácie, kde sú možné rôzne interpretácie beatu. Tieto situácie vznikajú kvôli tomu, že nikde nie je špecificky určený zvuk alebo udalosť, ktorým beat začína. Na to aby ich systém, takéto situácie zvládol, Masataka Goto a jeho tím použil agentov, ktorí sledujú a zaznamenávajú beaty z rôznych pohľadov. Každý z agentov má v sebe zadefinované isté bubenícke "patterny", ktoré porovnáva s práve detekovaným beatom a na základe podobnosti sleduje a zaznamenáva beaty podľa svojej vlastnej predstavy. Tiež si ukladá premennú,

ktorá znázorňuje spoľahlivosť zaznamenaných beatov, prípadne predpovedí beatov.

Tento systém pracoval v reálnom čase, ale ako si vieme predstaviť na tú dobu bolo bežanie veľkého množstva agentov veľmi náročné a preto bežal na paralelnom počítači FUJITSU AP1000. Pri experimentoch agenti používali 8 preddefinovaných bubnových “patternov” a testovali 42 populárnych pesničiek. BTS správne našiel beaty v 40, čo bol veľký úspech. Kedže tento systém bol jeden z prelomových, rozoberiem ho ešte detailnejšie.

BTS [20] je postavený na architektúre viacerých agentov, v ktorých každý zabezpečuje rôznu stratégiu pre zachytávanie beatu (obr. 3-1).



obrázok 3-1 Schéma BTS, [23]

Ako som spomíнал vyššie, to že tento systém používa viacerých agentov nám umožňuje zvládnuť rôzne situácie neprehľadné situácie. Aj keď jeden z agentov zle vyhodnotí nejakú situáciu alebo sa stratí, je veľká pravdepodobnosť, že BTS stále vráti priateľný výsledok. Ako vidíte na obrázku, každý z agentov vyhodnotí dátá získane z frekvenčnej analýzy, urobí hypotézu beatu podľa preddefinovaných modelov a vyhodnotí svoju vlastnú spoľahlivosť. Úplne na konci tohto procesu, kde každý z agentov bude ponúkať istú verziu toho, ako zachytil a pochopil beat, si “manager” vyberie najspoľahlivejšieho agenta (podľa premennej, ktorú si každý z nich ukladal).

Ďalšiou z dôležitých súčasti tohto systému je informácia o “drum patterne”. Každý z agentov obsahuje sebe vlastný pattern a s ním porovnáva práve prebiehajúci beat a na základe informácií z neho sa rozhoduje. Typický príklad takéhoto patternu je napríklad: basový bubon hrá vždy na výrazné (silné) doby a malý bubon hrá na nevýrazné (slabé) doby. Ďalší agent zas môže používať presne opačný systém: malý bubon na prvú a tretiu dobu (silné) a basový bubon na druhú a štvrtú (slabé). Ako vidno, na zadanie patternov agentov je nutné ovládať hudobnú teóriu a to bola aj jedna zo slabín tohto inak úspešného systému. Tiež samozrejme, ak bola skladba, v ktorej nemali bieť až taký podiel na vytváraní rytmu, alebo dokonca kde neboli vôbec, tento systém bol v podstate nefunkčný.

Ako bolo vidno na obrázku, ešte pred agentami sa nachádza frekvenčná analýza. Práve pomocou tej sa detekujú zmeny energií a na základe nich sa detekujú rôzne udalosti (nástup basového bubnu atď... ). Avšak keďže sa pracuje vždy s realistickým a tým pádom veľmi zložitým zvukovým signálom, môžu byť tieto udalosti aj nesprávne. Napríklad môžu omylom zachytiť basový bubon aj tam, kde nebude. Takáto nízko levelová analýza nemôže sama o sebe poskytnúť dostatočné informácie, aby sme vedeli s určitosťou povedať, že či išlo naozaj o basový bubon, alebo nie. Preto sa pri každej udalosti (evente) ukladá informácie o jeho spoločnosti, čím sa snažila táto chyba znížiť. Čím spoločnejšia bola udalosť, tým väčší význam bude mať vo výsledku BTS. Napríklad spoločnosť nástupu (čo je udalosť = event) sa vyhodnocuje pomocou faktoru, ako rýchlo stúpla energia za požadovanú hranicu (čím pomalšie, tým menšia spoločnosť) a tiež pomocou porovnávaní energií v jeho blízkosti. Ak sú v jeho blízkosti všetko veľké energie, ktoré sú veľmi blízko hranice zdetektovania beatu, spoločnosť bude nižšia. Spoločnosť hypotéz sa zasa vyvíja od toho, ako veľmi sa zhodujú preddefinované beaty s tými, ktoré sme zachytili vo frekvenčnej analýze.

Tiež je nutné spomenúť, že tento systém podporuje interakciu medzi “hládačmi nástupov” vo frekvenčnej analýze a agentami, ktoré tieto výsledky reprezentujú na vyšej úrovni. Najlepšie to ukážeme na príklade. Na nízkoúrovňovej frekvenčnej analýze máme viacero druhov hládačov nástupov. Jedni z nich hľadajú nástupy (rapídne zmeny energií) len na určitých frekvenčných

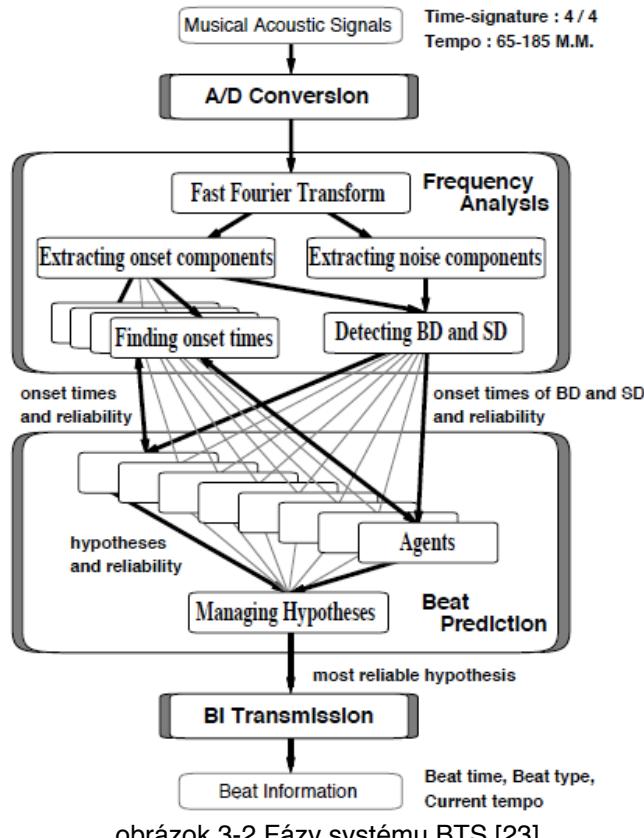
pásmach a iní sa zasa zameriavajú na nástupy s rôznou senzitivitou. Každý z týchto hľadačov spolupracuje s dvoma agentami, zvanými agentský pár (agent-pair). Každý pár dostane od jemu priradených hľadačov informácie o nástupoch a môže im na oplátku pozmeniť parametre. Napríklad keby agent dostával stále výsledky so zlou spoľahlivosťou agent mu upraví parametre, tak aby vyskúšal spracovať iné dátu. Napríklad by ho naviedol nech sleduje iné frekvenčné spektrum, alebo nech zachytáva aj nižšie zmeny energií. V tomto systéme existuje teda efektívna komunikácia medzi nízkou (frekvenčná analýza a hľadači v nej) a vyššou úrovňou (agenti).

### 3. 2. 1 Popis systému BTS

Systém bol navrhnutý tak, aby spracovával skladby v 4/4 takte (štyri štvrtové noty, osem osminových, šesťnásť šesťnástkových) a aby tempo bolo medzi 65 a 185 bpm (úderov za minútu). Tieto predpoklady spĺňa veľké percento hudby z populárnej scény. Tento systém sa zameriaval hlavne na pozície štvrtových nôt a zmeny tempa v podstate ignoroval, nebol navrhnutý tak, aby rátal s niečím takým, ako je zmena tempa počas skladby. Predpokladá, že tempo je konštanté. Taktiež, ako je spomínané vyššie, beat musia zabezpečovať bicie nástroje. BTS vysiela tzv. *BI* (beat information), v ktorom sa nachádzajú informácie o dočasnej pozícii beatu, o tom či je silný alebo slabý (výrazný, nevýrazný) a tiež tempo, v ktorom sa pracuje.

Dve hlavné fázy tohto systému môžme rozdeliť podľa [20] na: *frekvenčnú analýzu*, v ktorej sú detekované rôzne udalosti a *predvídanie beatu*, kde sa skúmajú rôzne hypotézy umiestenia beatu (vid' obrázok 3-2). Na úrovni frekvenčnej analýzy sa teda detekujú rozličné udalosti (eventy), ako napríklad nástupy dvoch hlavných zvukov podľa ktorých sa v tomto systéme detektuje beat: a to pomocou basového bubnu (BD = bass drum) a malého bubnu (SD = snare drum) a tiež nástupy na rôznych frekvenciách. Na vyššej úrovni predpovede beatov (beat prediction) tieto informácie spracovávajú a interpretujú agenti na základe preddefinovaných stratégii ako rôzne hypotézy s rôznou spoľahlivosťou. Každý agent najprv vypočítá interval medzi beatmi, potom začne predpokladať

kedy príde ďalší beat a nakoniec vyhodnotí spoločnosť svojej hypotézy o tom ako podľa neho vyzerá a znie beat. Následne je vygenerovaná informácia o beate (BI) na základe hypotézy s najväčšou spoločnosťou. Posledná fáza na obrázku je *BI Transmission* fáza, kde BTS posielá informáciu o beate do externej aplikácie, ktorá už informáciu o beate spracuje podľa vlastných potrieb.



obrázok 3-2 Fázy systému BTS [23]

Na výpočet frekvenčnej analýzy bola použitá rýchla fourierova transformácia (FFT). Viac informácií o nej môžete nájsť v kapitole 2. 2. 2. Vďaka FFT sme dosiahli isté frekvenčné spektrum, z ktorého môžme zísť rýchlosť zvýšenia energií. Tieto zvýšenia označujeme ako nástupy (onset component) a každý obsahuje ešte informáciu o stupni nástupu (ako rýchlo sa energia zvýšila). Ak teda komponent frekvenčného spektra spĺňa tieto podmienky (1), vtedy ho označujeme ako nástup.

$$\begin{cases} p(t, f) > pp \\ np > pp \end{cases} \quad (1)$$

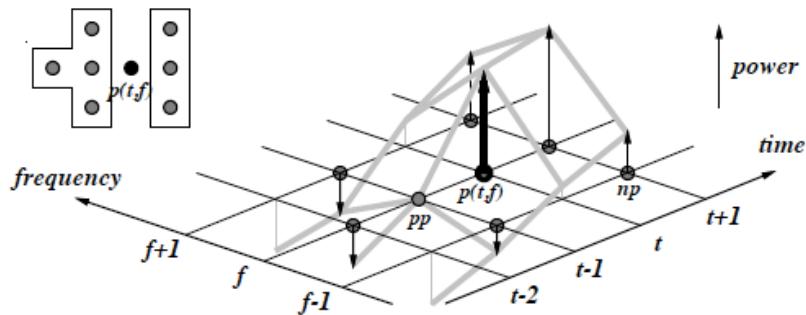
pričom  $p(t, f)$  je energia spektra na frekvencii  $f$  v čase  $t$  a  $pp$  a  $np$  sú zadané ako:

$$pp = \max(p(t-1, f), p(t-1, f \pm 1), p(t-2, f)) \quad (2)$$

$$np = \min(p(t+1, f), p(t+1, f \pm 1)) \quad (3)$$

Ak je  $p(t, f)$  naozaj nástup, jeho stupeň označujeme ako  $d(t, f)$  a je definovaný ako:

$$d(t, f) = \max(p(t, f), p(t+1, f) - pp) \quad (4)$$



obrázok 3-3 Detekcia nástupu [21]

Na obrázku 3-3 môžete vidieť ako vyzerá nástup, ktorý spĺňa všetky nutné podmienky. Energia v predchádzajúcich časoch a okolitých frekvenciach je menšia ako potencionálny nástup a energia nasledujúca v čase hned' za našim nástupom je tiež väčšia ako predchádzajúce takže sme úspešne vyextrahovali jeden nástup. Ďalej sa hľadajú časy nástupov. Túto funkciu spĺňajú hľadači, ktorí pracujú na rôznych frekvenciach (0 - 125 Hz, 125 - 250 Hz, 250 - 500 Hz, 500 Hz - 1 kHz, 1 - 2 kHz, 2 - 6 kHz a 6 - 11 kHz). Každý čas nástupu je daný časom, v ktorom nastane na danej frekvencii vrchol (peak). Nato sa využije výber vrcholu (peak-picking) v  $D(t)$ , kde  $D(t)$  je suma stupňov nástupov.

$$D(t) = \sum_f d(t, f)$$

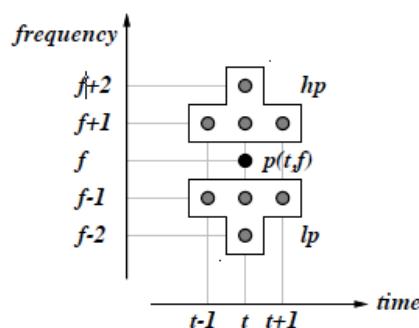
Znovu musíme vypočítať spoľahlivosť tohto času. To sa vyráta ako pomer hodnoty aktuálneho vrcholu s hodnotou maximálneho vrcholu (lokálneho). Každý hľadač má rôzne parametre, jeden z nich je napríklad frekvencia, pomocou ktorého môžeme nájsť časy nástupov v rôznych frekvenčných spektrách.

Z obrázku ďalej vidíme, že nasleduje časť: extrahovanie hluku. Tento krok je dôležitý hlavne pre detekovanie SD (malý bubon). V tejto časti sa predpokladá, že zvuky, ktoré nie sú hlukom majú rôzne harmonické štruktúry a ich energie sa vyvíja a mení, a tak môžeme komponenty, ktorých energia je v podstate konštantná na rôznych miestach, extrahovať a považovať ich za potencionálne zvuky malého bubna.

Znovu teda zadefinujeme  $n(t, f)$ , ktorého budeme považovať za potencionálny malý bubon (SD), ak spĺňa nasledovné podmienky:

$$\begin{aligned} hp &> p(t, f) / 2 \\ lp &> p(t, f) / 2 \end{aligned} \quad (6)$$

$$\begin{aligned} hp &= (p(t \pm 1, f + 1) + p(t, f + 1) + p(t, f + 2)) / 4 \\ lp &= (p(t \pm 1, f - 1) + p(t, f - 1) + p(t, f - 2)) / 4 \end{aligned} \quad (7)$$

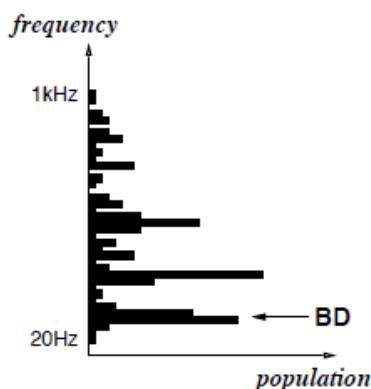


obrázok 3-4 Extrahovanie hluku (detekcia malého bubna) [20]

Na obrázku 3-4 vidíme, aké okolie berie do úvahy s porovnaním pri extrahovaní nástupu. Tam sa pozeralo predovšetkým naľavo a napravo po časovej osi, tu sa pozerá hlavne na frekvenčnú os.

Malý bubon je teda detekovaný z hlukových komponentov (noise components) a basový bubon je detekovaný z nástupov. Keďže zvuk basového bubnu nie je vopred známy, BTS sa ho naučí podľa danej pesničky pomocou nástupov. Pre časy, kde sa nájdu nástupy, sa BTS pozrie na všetky vrcholy na frekvenčnej osi a basovému bubnu určí najnižšiu nájdenú frekvenciu, keďže podľa názvu je najviac basový.

BTS potom detektuje basový bubon vtedy, keď' je nájdený nástup a frekvencia pri nástupe sa rovná premennej, v ktorej je uložená frekvencia basového bubna.



obrázok 3-5 Výber frekvencie basového bubna [21]

Ako vidíte na obrázku 3-5, systém detekoval nástup a zo všetkých frekvencií, kde došlo k nárastu energií vyberie najmenšiu a tú si zapamatá ako basový bubon.

### **3. 2. 2 Predpoved' beatu**

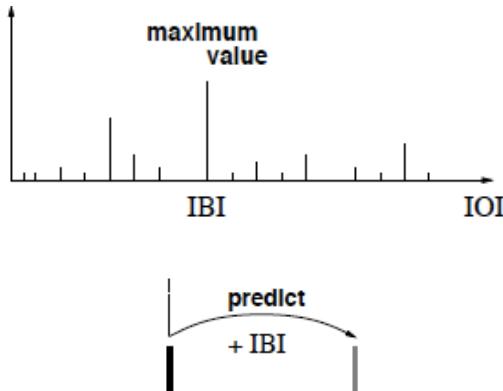
Na úspešné detekovanie a zaznamenávanie beatov v reálnom čase je nevyhnutné vedieť predpokladať ako asi sa beat vyvinie, pretože keď skončí spracovanie akustického signálu, jeho nástup už prešiel. Výsledky frekvenčnej analýzy spracovávajú agenti podľa svojich preddefinovaných stratégií a interpretujú ich ako hypotézy s určitou spoľahlivosťou. Každá tátó hypotéza obsahuje očakávaný čas, kedy príde ďalší beat, typ beatu, a aktuálny časový interval, ktorý je medzi beatmi. Hypotézy od všetkých agentov sú zbierané tzv. manažérom, ktorý vyberie najspoľahlivejšiu a pošle ju na výstup.

Všetci agenti sú rozdelení do párov. Každý pár agentov má spoločný interval medzi beatmi, čo umožňuje vyvarovať sa chyby ak sa jeden z agentov zle rozhodne a označuje beat presne medzi beatmi. Každý pár agentov je rôzny v tom, že príjma rozdielne informácie o nástupoch, pretože ma k sebe priradených iných “hlăadačov” času nástupov (onset-time finders).

Každý agent má tri parametre, ktoré definujú jeho stratégiu na vytvorenie hypotézy. Agenti, ktorí sú spolu v páre, majú rovnaké nastavenia týchto parametrov. Prvé dva parametre sú senzitivita a rozsah frekvencie, ktoré má agent skúmať. Tieto parametre kontrolujú korenšpondujúce parametre hlăadača nástupov a upravujú kvalitu a spoľahlivosť informácií, ktoré agent príjma. Pár agentov s vysokou senzitivitou majú väčšinou krátke intervaly medzi beatmi a naopak. Tretí parameter sa nazýva *stratégia histogramu* a nadobúda bud' hodnotu: *succesive* alebo *alternate*. Ak je hodnota *succesive*, na vytvorenie IOI (inter-onset interval - rozdiel medzi dvoma po sebe idúcimi nástupmi) histogramu sa použijú po sebe idúce nástupy, inak sa použijú hodnoty striedajúcich sa nástupov. [20]

Hypotézy sú teda veľmi dôležitým výsledkom práce agentov. Všetko začína tak, že každý agent najprv vypočíta IBI (inter beat interval - vzdialenosť medzi beatmi), predurčí čas, kedy príde ďalší beat a na záver vyhodnotí svoju vlastnú spoľahlivosť. Ďalej agent určí typ detekovaného beatu a na základe tejto novej informácie zmodifikuje svoju spoľahlivosť. Ak má agent dlhodobo nízku spoľahlivosť dokáže si zmeniť svoje parametre. Na záver manažér vyberie najspoľahlivejšiu hypotézu.

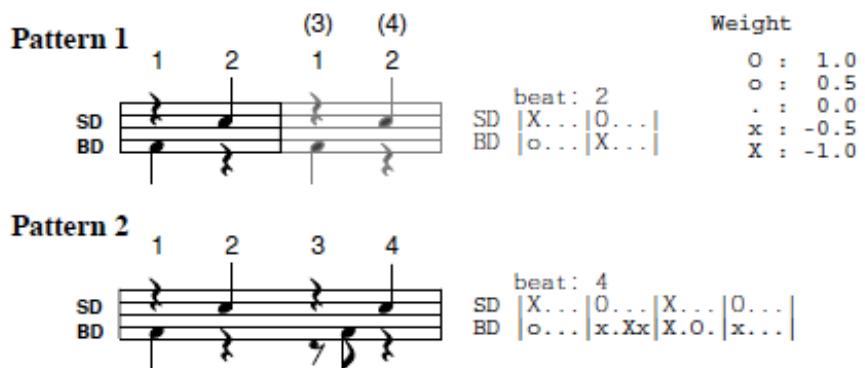
Najprv si ukážeme, ako agenti vyrávajú a odhadujú čas, kedy príde nový beat. Jednoducho prirátajú k času, v ktorom zaznamenali beat svoj IBI (inter beat interval). IBI sa vyráta z IOI histogramu (inter onset interval) ako maximálna hodnota nameraná v tomto histograme na obrázku 3-6. V jeho hornej časti môžete vidieť IOI histogram a pod ním spôsob vypočítana predpovede beatu.



obrázok 3-6 IOI histogram (hore), spôsob vypočítania predpovede beatu(dole).[20]

Ako som už často spomínał, každý agent vyhodnotí svoju vlastnú spoľahlivosť. Ak detekovaný beat zhoduje s preddefinovanými beatmi je spoľahlivosť svojej hypotézy sa zvýši (zvýši kvôli tomu, že to nie je jediná vec, ktorá ovplyvňuje agentovu spoľahlivosť), ak sa zhoduje s nejakou výchylkou, napr. osminová alebo šestnástková nota, spoľahlivosť sa zvýši menšou mierou. Ak sa nezhodujú vôbec, spoľahlivosť hypotézy sa zníži.

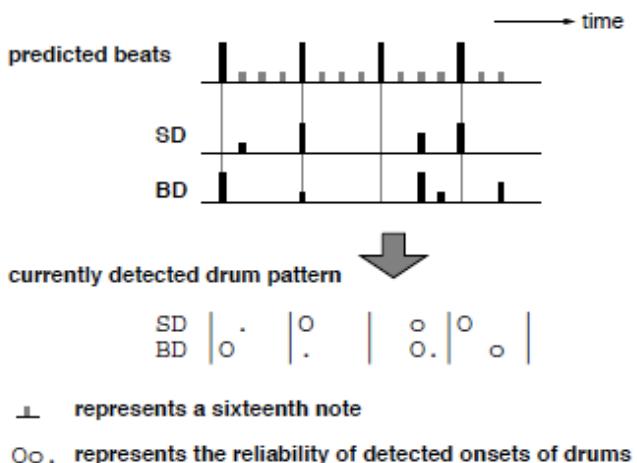
Na obrázku 3-7 môžete vidieť dva príklady preddefinovaných bubenových vzorov, ktoré agenti porovnávajú s detekovanými beatmi.



obrázok 3-7 Forma preddefinovaných bubenových vzorov [21]

Tieto vzory ukazujú typické používanie basového a malého bubna v popovej a rockovej hudbe. Vzor by mal začínať vždy silnou dobou ale nemusí to tak vždy byť. Dĺžka vzora je buď celý tak ako v Patterne 2 alebo jeho polovica ako v Patterne 1. V tom prípade sa jednoducho polovica zopakuje a vytvorí sa tak vzor na celý takt (stále uvažujeme, že na vstup BTS dostane vždy skladbu v 4/4 takte).

Typ beatu a jeho spoľahlivosť si vyrátame pomocou obrázku 3-8.



obrázok 3-8 Porovnávanie detekovaného a preddefinovaného beatu [20]

Nástupy bicích sú teda formované do práve detekovaného vzoru. Všetky vzory sú uložené s presnosťou na jednu šesťnástinovú notu, to znamená, že v 4/4 takte je 16 voľných miest, kde sa môže vyskytnúť alebo nevyskytnúť nota. Tento výsledný vzorec, môže vyzeráť ako ten na obrázku. Tu tiež môžeme vidieť, aký veľký význam má parameter spoľahlivosti pri nástupoch. Vidíme, že systém zachytil istý nástup malého bubnu už v prvom takte na pozícii druhej šestnástky, ale keďže ma malú spoľahlivosť, v podstate tento nástup neberieme veľmi vážne. Tento náš novo-získaný vzor potom porovnávame s preddefinovanými vzormi, na základe tohto porovnania získame pre každý uložený vzor nejakú zhodu. Typ beatu, to znamená, že ktorá časť beatu bude silná a ktorá slabá sa okopíruje zo vzoru, s ktorým je zhoda čo najväčšia. Ak je zhoda vysoká, v tom prípade je veľká aj spoľahlivosť. Vtedy sa stáva, že IBI (inter beat interval) je v podstate štvrtová nota a skoro vždy je vybraná táto hypotéza (je to v podstate najlepší prípad, aký môžeme dostať).

Ešte sme spomenuli problém, keď je dlhodobo spoľahlivosť hypotézy od agenta nízka. Vtedy sa agent rozhodne, že dané parametre nie sú vhodné pre daný vstup a upraví si ich (s ohľadom na ostatných agentov). Tento proces funguje v dvoch krokoch: Majme dvoch agentov *A* a *B*. Agent *A* vykazuje už dlhý čas (vopred zadaná premenná) nízku spoľahlivosť a tak si od manažéra vypýta povolenie na zmenu svojich parametrov. Je potrebné ešte spomenúť, že agenti *A*

a  $B$  spolu tvoria pár. Ak je spoločnosť agenta  $B$  vysoká (respektíve nízka), manažér zakáže agentovi  $A$  zmenu parametrov. Ak ju už povolí, zakáže ostatným agentom meniť premenné na istý čas. Agent  $A$  teda dostane povolenie na zmenu parametrov a tak si vyberie novú množinu troch parametrov, ktoré definujú jeho stratégiu, na základe ktorej dedukuje svoju hypotézu.

V celom tomto systéme je veľmi dôležitá funkcia manažéra. Ten všetky agentami vygenerované hypotézy rozdeľuje do skupín podľa IBI (inter beat interval) a času beatu. Každá skupina ma celkovú spoločnosť (suma spoločnosťí hypotéz). Výstupom je najspoločnejšia hypotéza v najspoločnejšej skupine.

### **3. 2. 3 Výsledky BTS**

Tento systém testovali na 42 populárnych skladbách rockovej a popovej hudby. Vstupom bol audio signál z kompaktného disku a vo všetkých skladbách boli hlavným nosičom beatu bicie nástroje. Tempo skladieb bolo v rozpätí 78 bpm až 184 bpm a bolo v podstate konštantné. Agenti dostali 8 preddefinovaných bubeníckych vzorov, ktoré porovnávali s práve detekovanými vzormi. BTS správne označilo beaty u 40 zo 42 pesničiek. Na začiatku každej pesničky, bol typ beatu (slabý alebo silný) označený vždy zle. Bolo to z toho dôvodu, že systém nemal na začiatku zadefinovaný frekvenciu basového a malého bubnu. Po pár taktoch, keď obidva bubny zneli rovnako po dlhšiu dobu, BTS upravil aj túto informáciu [20].

## 4. Informačná hodnota v hudbe (prieskum)

Ako sme si ukázali v predchádzajúcej kapitole, najviac používané techniky na vizualizáciu hudby nemajú v skutočnosti s hudbou až tak veľa spoločné. Dokážeme v nich pozorovať zmeny energií v určitých frekvenciách, ale vonkoncom neviem len vďaka týmto dátam zistiť niečo viac o hudbe, ktorá sa vizualizuje. Z týchto dát nevieme priamo zistiť charakter skladby, jej tóninu ani iné informácie. Samozrejme, pre veľké množstvo používateľov vizualizácií, ktoré ich používajú len ako vizuálny doplnok sú tieto informácie zbytočné.

Na zamyslenie však ostáva, čo môže očakávať od vizualizácie hudby človek, ktorý by tieto informácie mohol využiť. Položil som preto viaceré otázky na tému "Ako by vyzerala podľa vás vaša vysnená vizualizácia hudby" viacerým ľuďom, ktorí sa hudbe venujú na profesionálnej úrovni už dlhší čas. Ľudí som vyberal z rôznych oblastí a žánrov, aby bol zahrnutý čo najväčší okruh záujmov. Tiež som položil potom tie isté otázky ľuďom bez hudobného vzdelania a hudobných aktivít, aby som zistil, ako by chceli vnímať vizualizáciu hudby oni.

*Vladislav Šarišský : skladateľ, klavirista, (Oskar Rózsa, Jana Kirschner)*

*Marián Slávka : bubeník (IMT Smile, Andrej Šeban, Dlhé Diely ...)*

*Martin Žiak: basgitarista (Nikki tresor, Kubiz ... )*

*Andrej Hruška: producent, gitarista (Tina, Sanyland,...)*

*Emília Kormaníková: študentka Žurnalistiky na UK*

*Igor Rjabinin : študent Aplikovanej informatiky na UK*

*Lukáš Gejdoš: študent Aplikovanej informatiky na UK*

*Jozef Tomanica: študent Religionistiky na UK*

Ďalej som tieto otázky položil ľuďom z celého sveta pomocou fóra Mac-BB.org vďaka službe [www.survey-monkey.com](http://www.survey-monkey.com). Celkovo vyplnilo túto anketu presne 78 ľudí.

## 4. 1 Vzor otázok

1. Využívate možnosť vizualizácie hudby v hudobných prehrávačoch ?
  2. Ste spokojný s tým, ako daná vizualizácia zodpovedá danej hudbe?
  3. Ako by vyzerala podľa vás dokonalá vizualizácia hudby?
  4. Viete si predstaviť, že by vizualizácia obsahovala aj nejaké informácie?
  5. Aké informácie by ste chceli získat a využiť pri vizualizácii? (tónina, takt, charakter skladby)?
- 
1. Are you using visualization feature in your audio players?
  2. Are you satisfied how visualizer responds to actual music?
  3. How would you describe perfect music visualization?
  4. Can you imagine music visualization with some information?
  5. What kind of information would you like to use with visualization?

## 4. 2 Výsledky

Z prieskumu vyplýva, že 73 percent opýtaných nepoužíva pri hudbe vizualizačný softvér, 21.3 percenta ho niekedy v minulosti používalo a zvyšných 5.7 percenta ho používa občas - nikto zo 78 ľudí neuviedol, že vizualizér používa pravidelne. Na druhú otázkou 53.3 percenta respondentov odpovedalo, že sú so zodpovedaním vizualizérov s hudbou spokojný, zvyšné percentá tvorili ľudia, ktorí odpovedali nie. Pri opise perfektnej vizualizácií neboli na výber žiadne možnosti, takže odpovede skúsim zhrnúť do skupín. Jedna skupina respondentov tvrdila, že vizualizácie hudby sú stratou času, že hudba sama o sebe je veľmi silné médium. Ďalší považovali za dokonalosť vizualizáciu, ktorá by plynule s hudbou vytvárala nekonfliktný, vizuálne príťažlivý podklad. Profesionálny hudobníci na túto otázkou väčšinou odpovedali, že dokonalá vizualizácia sú, alebo by mala vyzeráť podobne ako noty. Na štvrtú otázkou odpovedalo 64,3 percenta "áno" a 35,7 percentia "nie". Posledná otázka znova nemala možnosti, takže ich znova rozdelím do skupín. Jedna zo skupín tvrdila, že je to strata času, ďalšia chcela, aby boli zakomponované údaje o autorovi, skladbe a albume. Profesionálny muzikanti sa vyjadrovali o informáciach ako tónina, harmónie, stupnice, bpm, akordy.

## 5. Analýza hudby

Veľkým rozdielom v získavaní týchto informácií je zdroj, z ktorých ho budeme získavať. Ak si zoberieme ako zdroj napríklad MIDI, tak by nebol až taký problém všetky tieto informácie pomocou jednoduchých algoritmov zistieť. MIDI je totiž taká špecifikácia, ktoré obsahuje informácie o trvaní tónu, jeho výške, nástroji, ktorý ho hrá. Na základe MIDI teda vieme získat úplne presné informácie o hudbe, ktorá znie.

Problém je, že vizualizácia MIDI súborov nie je presne to, čo sa nás týka, pretože málo kto si predstaví pod pojmom MIDI hudbu. Využitie MIDI vizualizácie však netreba úplne zatracovať práve kvôli tejto ľahko extrahovateľnej informácii. Každopádne ako samostatná vizualizácia by to nebol dokonalý systém, pretože nie ku každej hudbe máme dostupnú MIDI špecifikáciu, ale napríklad pre hudobníkov, ktorý počas koncertu používajú MIDI controller by bol takýto doplnok určite veľmi prijateľný.

Analýza hudobného obsahu z audio signálu je však oveľa komplikovanejší a náročnejší proces. Mnohé techniky, ktoré su uvedené aj v predchádzajúcich kapitolách sa totižto viac sústredia na samotný zvuk ako na hudbu. Aj v minulosti však vzniklo viac systémov, ktoré sa sústredili najmä na analýzu a následne na resyntetizáciu hudby. Jeden z prvých takýchto systémov je od Xaviera Serra z roku 1989 a nazýva sa *The Spectral Modeling Synthesis* (SMS). [27][28]. Tento systém pracoval s monofónnym(jedno-hlasný) a polyfónnym(viac-hlasný) signálom, tento signál sa snažil rozložiť na dve zložky - deterministickú a stochaistickú. Tento softvér mal viaceré použitia, ako napríklad zmena výšky tónu, kompresia signálu, analýza obsahu signálu, získanie typu nástroja, ktorý hrá a jeho morfovanie do iného.

Neskôr vznikli mnohé iné systémy, ktoré sa sústredili najmä na generovanie hudby. Jedny z najpoužívanejších metód na analýzu a resyntetizáciu sa stali také, ktoré sa sústredili priamo na to, ako sa hudba príjma. To znamená, že úloha tejto

metódy bolo prechádzať zvukovú databázu "samplov", porovnávať ich a tie čo sa najviac podobali spájať do celku tak, aby zhoda so zdrojovým signálom bola čo najväčšia. Tieto metódy boli v začiatku využívané ako súčasť TTS (text to speech) systému, kde prehľadávali veľké databázy slov a znakov a následne z nich konštruovali celé vety.

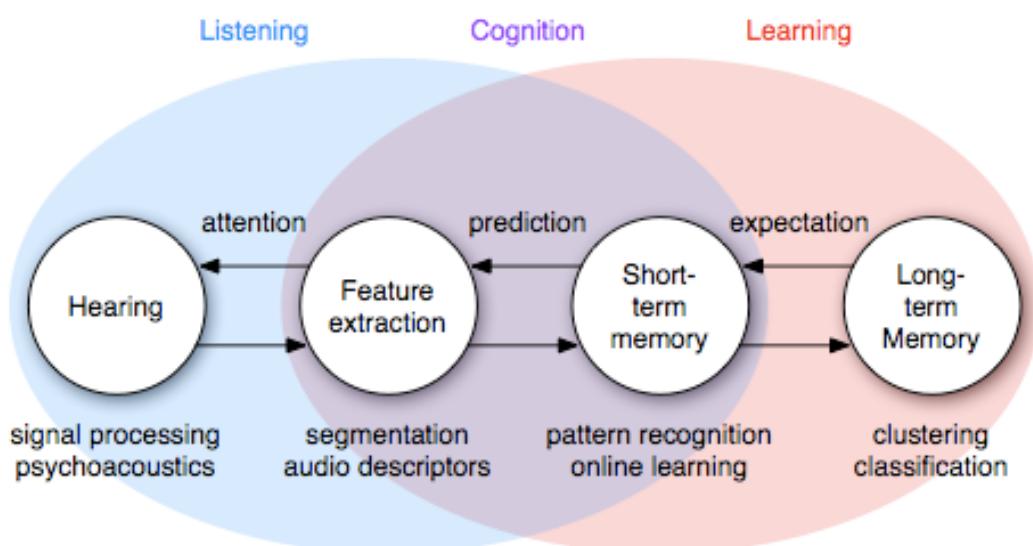
Všetky tieto systémy sa však sústredili najmä na resyntetizáciu hudby, zmena prišla s veľkým nástupom internetu, P2P (peer 2 peer) sietí a online hudobného trhu. Vtedy sa stalo veľmi dôležitým aspektom spôsob, akým usporiadaváme a manažujeme hudbu. Tak vznikla téma s názvom *Music Information Retrieval*, ktorá opisuje digitálny signál priateľnejšími a konkrétnejšími slovami. Vznikli tak takzvané metadáta (dáta o dátach), ktoré sú často pripojené k signálu. V nich sa však často nachádzajú údaje, ktoré nie sú automatické a treba ich manuálne zadávať. Vďaka potrebe získať tieto informácie však vznikli niektoré techniky, ktoré sú použiteľné v získavaní informácií o samotnom signáli. Jedna z nich je napríklad metóda *Thumbnailing*. Tá funguje na princípe odstraňovania menej dôležitých sekcií signálu a vybratia hlavnej sekcie, ktorá je pre daný signál (skladbu) charakteristický. Väčšinou ide teda o refrén, jeden z prístupov spracovania tejto metódy predstavuje Masataka Goto. [22].

Výskum v tejto oblasti tiež objasnil, že samotná hudba nemôže byť charakterizovaná úplne objektívne. Hudba totiž prináša veľmi veľké množstvo subjektívnych pocitov a hodnôt, ktoré sa časom menia a nedajú sa efektívne zachytiť.

Väčšina systémov pracuje na nasledovnom princípe, zo signálu pomocou analýzy ako FFT, Beat Detection získame dáta, ktoré istým spôsobom spracujeme a prezentujeme ich buď vizuálne, alebo ich použijeme na vytvorenie nového transformovaného zvuku (ako v prípade systémov spomenutých vyššie). To, o čo sa pokúšame poukázať v tejto práci je zahrnuté do tejto analýzy aj samotnú hudbu. Jedna z najdôležitejších fáz, ktoré teda musíme zvládnutie je fáza počúvanie signálu, presnejšie už hudby. Jeden z najväčších pokrokov v tejto oblasti spravili na univerzite MIT v sekcii Media Labs, kde Tristan Jehan vypracoval dizertačnú

prácu na tému Creating Music by Listening [30]. Svoj framework orientoval hlavne na vytváranie hudby a práve na Music Information Retrieval, na základe čoho založil spolu s kolegom Brianom Whitmanom spoločnosť EchoNest, ktorá je momentálne so svojím softvérom Musical Brain jedna z najlepších v danej oblasti.

Základom tohto systému je to, že k systému ako ho prezentoval Eric D. Scheier [she1] pridal učenie.



obrázok 5-1 Schéma systému na vnímanie hudby [30]

Ako vidíte na obrázku 5-1 toto je systém, ktorý nám poslúži na pochopenie hudby. Informácie prechádzaju z ľava do prava a pri každom prechode sa dátá zredukujú na jednoduchšie a viac abstraktné. Prvé tri bloky môžme nazvať počúvaním, pričom posledné tri zodpovedajú učeniu. Prenik týchto dvoch oblastí sa nazýva hudobná kognícia, kde sa dejú najzaujímavejšie veci v oblasti hudby.

V prvej oblasti je výsledkom signál, ktorý reprezentuje, to čo počujeme. V skutočnosti je ucho totiž limitované a len časť originálneho signálu vieme zachytiť. To znamená, že je zbytočné spracovávať celý signál. Výsledný signál má formu spektrogramu, ktorý znázorňuje len to, čo počuje naše ucho. V druhej fáze sa tento výstupný signál konvertuje do symbolickej reprezentácie, ktorá reprezentuje už samotnú hudbu (nazýva sa aj "Musical DNA"). Popisujeme tu teda signál vo

forme hudobnej plochy a výstupom je oveľa kompaktejšia charakterizácia hudobného obsahu. V tretej fáze (krátkodobá pamäť) sa analyzuje práve "hudobná DNA". Detekujú sa tu vzorce, na základe ktorých vieme predpovedať ako sa budú vyvíjať napríklad beaty atď. Posledná fáza zhlukuje informácie a klasifikuje výsledky pre dlhodobé učenie (úložná pamäť).

Pre kvalitné fungovanie systému musia všetky fázy medzi sebou komunikovať. Najmä je dôležité, aby komunikovala krátkodobá pamäť (rozpoznávanie vzorcov, beatov) s dlhodobou pamäťou, pretože tým simulujeme ľudské vnímanie hudby, ktoré tiež súvisí s dlhodobou a krátkodobou pamäťou.

Tento systém využíva vlastný sluchový model, ktorý vyplýva z toho, že sa snaží odstrániť informácie, ktoré sú nepotrebné a človek ich svojím obmedzneným sluchovým nástrojom aj tak nepríjme. Dobrým príkladom je napríklad MPEG audio layer 3 kodek (MP3), ktorý na podobnom princípe znižuje veľkosť audio súborov. Výsledkom tohto sluchového modelu je potom tzv "audio surface", ktorého grafická reprezentácia sa nazýva *auditory spectrogram* a presne znázorňuje to, čo človek dokáže počuť. Pomocou tohto spektrogramu teda presne rozumieme a vidíme zvuk z nášho zdroja, avšak stále nie hudbu. Z tohto spektrogramu vieme napríklad získať celkovú hlasitosť, respektíve intenzitu signálu. A to jednoduchým sčítaním amplitúd na všetkých frekvenčných rozhraniach.

$$L_{dB}(t) = \frac{\sum_{k=1}^N E_k(t)}{N}$$

kde  $E_k$  je amplitúda na frekvenčnom rozhraní  $k$  z celkového počtu  $N$  rozhraní spektrogramu. Samozrejme, existujú aj podrobnejšie modely získavania celkovej intenzity signálu, ako napríklad od Moorea a Glasberga v [32].

Ďalej skúsme už získať informáciu, ktorá sa viac blíži k hudobnej oblasti. Budeme sa teda snažiť získať informáciu o sfarbení tónu. Tento pojem sa tiež zvykne pomenovať cudzím slovom *timbre*. Táto informácia je dôležitá práve kvôli tomu, že vďaka nej vieme odlíšiť dva zvuky majúcu rovnakú hlasitosť aj výšku. V praxi táto informácia definuje hudobné nástroje. Už v roku 1978 J. Grey v

[33] tvrdil, že sa dajú rôzne sfarbenia nástrojov v orchestri majú pre seba charakteristické spektrálne plynutie a kvalitu nástupu (rýchlosť, intenzita). Keďže však veľké množstvo dát odstráníme v prvej časti, pri vytváraní nášho hudobného priestoru, ktorý zodpovedá nášmu uchu, nemôžeme tieto charakteristiky efektívne využiť.

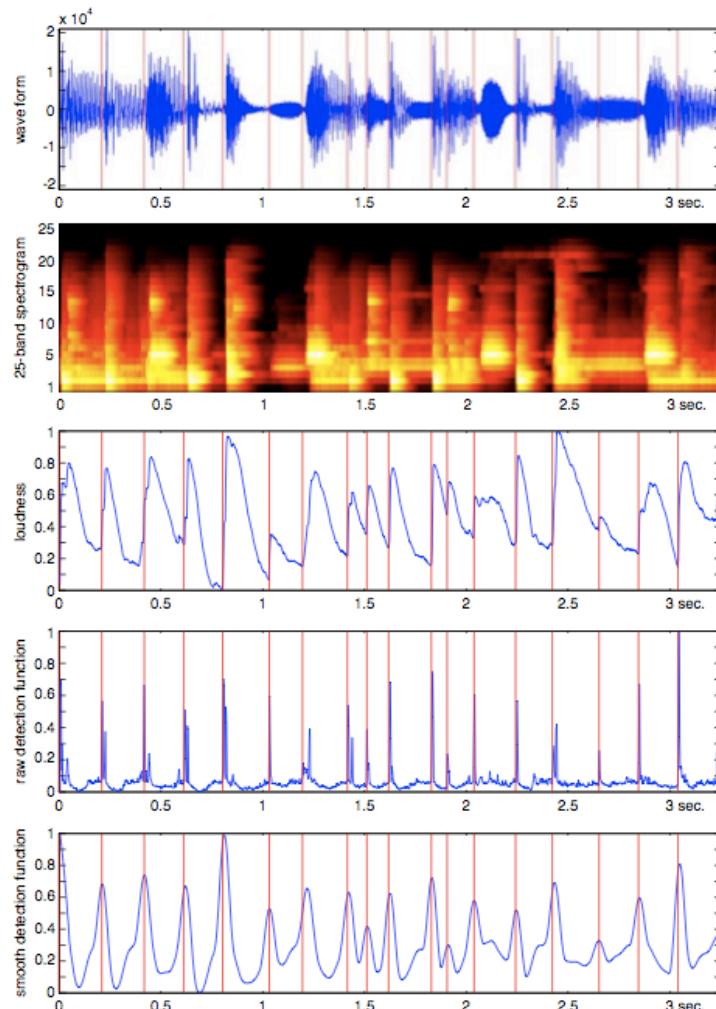
Preto je v dnešnej dobe trendom používať na charakterizáciu farby a viacerých hudobných atribútov nízko-levelové zvukové "deskriptory" (LLD - low level audio descriptors). Tie sú organizované do rôznych kategórií ako napríklad: energitické (sledujú a merajú zmeny energií signálu), spektrálne (vypočítavané z fourierovej transformácie), harmonické (vypočítane z sínusoidného harmonického modelovania [34].) Ďalším krokom je nájsť takú kombináciu týchto LLD deskriptorov, ktoré budú čo najlepšie súhlasiť s našim cieľom.

Ešte výhodnejším riešením sá zdá reprezentovať každé sfarbenie jednoducho porovnatelnou reprezentáciou. Podľa psychoakustikov je pre ľudské ucho kriticky dôležitých len 25 frekvenčných rozsahov a vďaka nim vieme pokryť charakteristiku celého spektra. Takže ich môžme považovať opodstatnené kvalitný popis jednotlivého zafarbenia.

Ďalšia dôležitá informácia je segmentácia, alebo detekovanie nástupov. Znamená to, že vieme rozdeliť audio signál na menšie jednotky. Nás keďže pracujeme s hudbou zaujíma delenie len po najmenšie rytmické oblasti, ktoré sa v hudbe objavujú (individuálne noty, akordy atď.). Každý segment, na ktorý budeme signál deliť definujeme pomocou dvoch vlastností: jeho nástupu a jeho hraničných oblastí. V každom tomto segmente by nemalo dojsť k zásadnej a prudkej zmene zafarbenia, avšak pri nástupe segmentu by mal práve takáto prudká zmena nastať. Ņou vlastne deketujeme samotný nástup nového segmentu [31].

Náš získaný spektrogram si skonvertujeme na funkciu detekujúcu udalosti. Výsledný signál ukáže výkyvy, ktoré budú korenšpondovať nástupom. Následnou filtriaciou získame vyhľadenú funkciu, ktorá je vhodná pre štádium, kde budeme vyberať vrcholy signálu. Túto vyhľadenú funkciu porovnáme ešte s hlasitosťou

funkciou, pretože každý nástup v hudbe je väčšinou doprevádzaný zmenou hlastnosti (najčastejšie smerom nahor). Aby sme zachytili rýchlosť nástupu, na predchádzajúcom malom časovom úseku (väčšinou okolo 20 milisekúnd) nájdeme lokálne hlasitostné minimum, ktoré zodpovedá najtichšiemu momentu pred nástupom a to je to pravé miesto kde môžme zadefinovať začiatok nástupu. Na obrázku 5-2 je zaznamenaný 3.25 sekundy dlhý úsek zo skladby "Watermelon man" od Herbieho Hancocka. Na prvom obrázku môžete vidieť typickú vlnu modrej farby, pričom červenou sú označené nástupy. Ďalej nasleduje typický spektrogram tohto zvukového systému zredukovaný na 25 pre ľudské ucho kritických frekvenčných rozhraní. Pod ňou vidno hlasitostnú funkciu a následne detekujúce funkcie (najprv bez a následne s vyhladením). Ako prah vyberieme lokálne maximum z vyhladenej funkcie (ako môžete vidieť na úplne dole na obrázku 5-2).



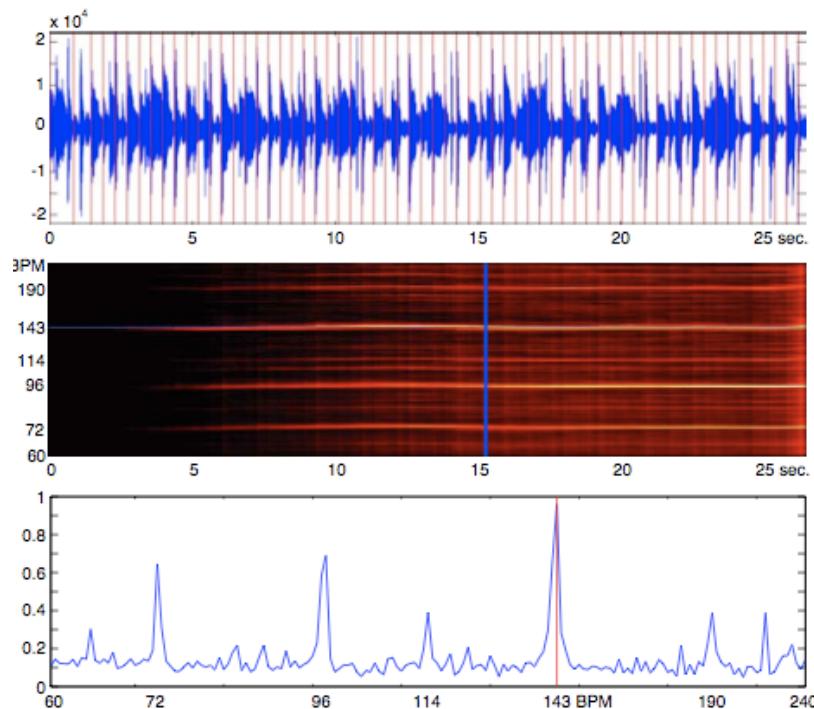
obrázok 5-2 Detekcia udalostí pomocou spektrogramu. Zhora dole: zvuková vlna, spektrogram, hlasitostná funkcia, funkcia detekovania, vyhladená funkcia detekovania [31]

Ďalším dôležitým pojmom v oblasti z hudobného vnímania je beat a tempo. Kedže tejto téme som sa venoval dostatočne v predchádzajúcej kapitole, spomienom len v skratke v akej modifikácii sa používa v tomto systéme, keďže samotné informácie o beate sú dôležité aj pri extrahovaní využiteľnej informácie napríklad pre bubeníkov (typ taktu, tempo atď). Aj keď používateľ by musel rátat s chybovosťou a nedokonalosťou beat detection techník.

V tomto systéme sa teda využíva znova násť zredukovaný 25 frekvenčný spektrogram a Scheirov systém rezonátorov [29]. Tempo z nieho vieme získať nasledovným spôsobom. Rozsah od 60 do 240 bpm (úderov za minútu) je rozložený na veľkú množinu filtrov, ktorých vlastnosťou je že pri určitom tempe majú rezonovať. Filtry sú testované na viacerých frekvenciach súčasne a sú nastavené aby existovali len pár sekúnd (pracuje sa tu len s krátkodobou pamäťou). Tempo spektrogram získaný týmito filtrami sa potom vyráta ako súčet interných energií. Môžete ho vidieť na obrázku 5-3. Ako môžete vidieť výsledné tempo určil spektrogram ako 143 bpm, vybral ho ako vrchol zo spektogramu. Práve tento výber je jedna z hlavných nevýhod danej techniky, pretože takýto výber nie je spoľahlivý. V strede vidíte tempogram, ktorý na začiatku nemá žiadne vedomosti, ale postupne získava predstavu o tempe, naopak v hornom paneli máme informáciu o detekcii beatu hned po prvej sekunde. Na dolnom paneli môžeme vidieť konkrétny výstup každého rezonátora. Najvyšší vrchol je naše tempo.

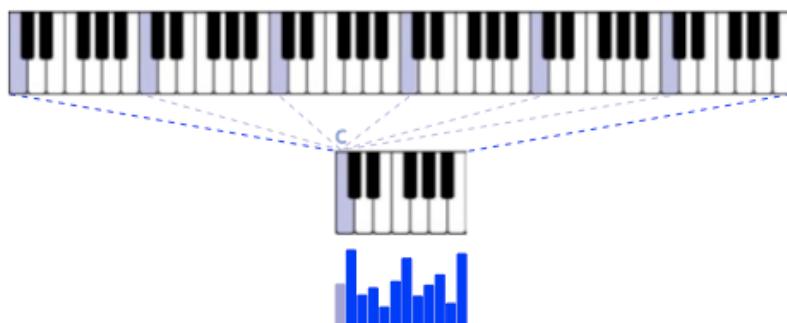
Ďalšia a asi najdôležitejšia informácia z hľadiska využiteľnosti je informácia o výške tónu a harmónii. V tejto sekcii sa využijú aj informácie získané pri segmentácií. Segmenty totiž obsahujú a reprezentujú individuálne noty, akordy, zvuky bubenov alebo oblasti s rovnakým farebným alebo harmonickým charakterom. Ak by bola segmentácia perfektná, v samotnom segmente nenastane žiadna prudká zmena vo výške alebo farby tónu. V tejto sekcii budeme riešiť práve sledovanie výšky tónu. Signál môže byť rôzny a podľa toho sa mení aj náročnosť tejto sekcie. Napríklad problém sledovania výšky tónu v polyfónnom signáli stále ešte iba čaká na vyriešenie, hlavne v zmiešanine zvukov, ktorá obsahuje aj bubny. Na zjednodušenie teda použijeme definíciu z [35] a pojem

*chroma*. Na rozdiel od klasickej výšky tónu, ktorá rozdeľuje tóny aj podľa oktáv, chróma je cyklická a nerozlišuje oktávy. Vo formulácii chromy sú teda napríklad tóny C a c1 rovnaké aj keď v skutočnosti sú medzi nimi tri oktávy.



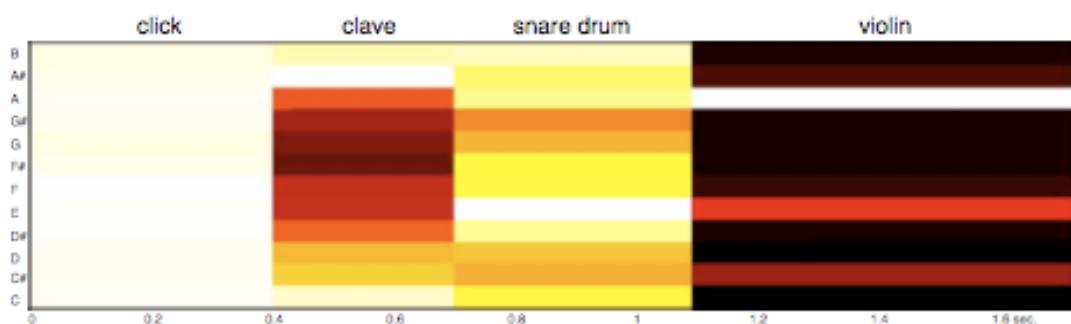
obrázok 5-3 Vypočítanie tempa zo spektrogramu [31]

Pomocou FFT získame dostatočný frekvenčný obsah, ktorý použijeme na naplnenie nášho 12 - dimenzionálneho *chroma* vektora (obrázok 5-4). Celé spektrum (od C1 na frekvencii 65 Hz až po H7 na frekvencii 7902 Hz) teda musíme premapovať na 12 rôznych výšok tónov. Táto aproximácia je využiteľná a aj keď stratíme veľké množstvo dát, na pochopenie základných harmonických charakteristík to však stačí. Vznikde nám teda vlastne dvanásť tónová chromatická stupnica, ktorá sa najviac hodí na európsku hudbu, avšak s istými odlišnosťami by sme ju vedeli aplikovať aj na stupnice indické, čínske, arabské a iné. Výsledný *chroma* vektor následne ešte normalizujeme a to tak, že každý jeho element vydelíme elementom s maximálnou hodnotou, čím vlastne element s maximom sa bude rovnať jednotke. Reprezentácia tohto vektora s ohľadom na čas sa nazýva chromogram.



obrázok 5-4 12 - dimenzionálny vektor chroma [30]

Takýto chromogram vieme zestrojiť pre každý segment, takže pri perfektnej segmentácii vieme predpokladať, že v každom segmente bude nástroj s jednotným zafarbením ( jeden segment - jeden nástroj). Chromogramy rôznych segmentov môžu vyzeráť napríklad tak ako na obrázku 5-5. Zvuk úplne naľavo je digitálny klik, ktorý sa používa v metronónoch, vedľa sú drevené perkusie (claves), ďalej rytmičák (malý bubon) a nakonci husle. Čím väčšiu energiu daný element (tón) má, tým svetlejšie je znázornený. Preto vidíme, že napríklad digitálny klik z metronomu sa najčažšie špecifikuje, lebo je energeticky plochý. Najlepšie a najčahšie harmonicky rozlíšiteľné sú husle, kde vidíme že tón, ktorý sa s najväčšou pravdepodobnosťou hrá je A.



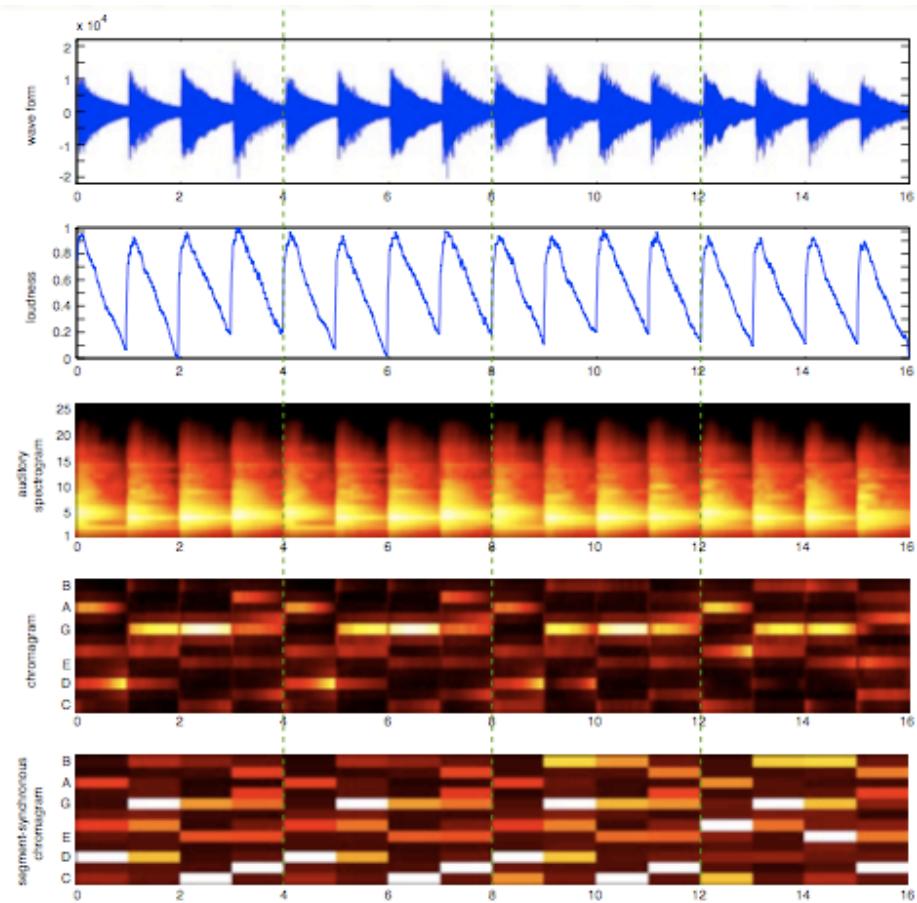
obrázok 5-5 Chromogramy rôznych segmentov(nástrojov), [30]

Môžme teda vidieť, že v našej reprezentácii budeme vypočítavať chroma vektor pre každý segment. Tento prístup nie je veľmi zvyčajný ale má viaceré výhody. Niektoré z nich sú napríklad efektívna reprezentácia pre budúce využitie alebo znížené výpočtové nároky (FFT vypočítavame pre každý segment oddeleno, to znamená, že nebojujeme s problémom, keď sa fourierove transformácie prekrývajú).

Na obrázku 5-6 môžme vidieť demonštráciu výhod našej reprezentácie výšky tónu v prípade ak hudba obsahuje len jeden nástroj. V tomto prípade hral klavír typický postup II-V-I (presnejšie akordy D7-G7-C7), ako vidíme z prvých troch panelov (vlnenie, hlasitostná krivka, spektrogram) nevidíme žiadne zmeny a mohli by sme si myslieť, že sa hrá stále ten istý akord dookola. Je to spôsobené tým, že daný signál obsahuje len jeden nástroj. V posledných paneloch (chromogramy) však už vidíme harmonické zmeny. V prvom to nie je také výrazné, pretože ten je vypočítavaný každých 6 milisekúnd na 93 milisekúnd dlhom okne, na rozdiel od posledného, ktorý je získaný našou technikou (jednen chroma vektor na jeden segment) vidíme harmonické zmeny oveľa krajšie. Dokonca môžme vidieť okrem hlavného tónu akordu aj iné intervale, napríklad pri druhom akorde vidíme, že najintenzívnejší je tón D (prima) a dokonca vidíme že celkom vysokú energiu má aj tón F, čo znamená, že znie interval malá tercia, čo znamená, že akord, ktorý znie je molový. Tým vlastne vidíme, že takáta reprezentácia vektoru chroma (na segmenty) je aj jednoduchšia ale pritom stále udržiava dôležité informácie. Samozrejme, to aké tóny sú detektované stále záleží od energií na daných frekvenciách, takže výsledok nemusí byť vždy korektný. Ak však predpokladáme kvalitnú segmentáciu miera chyby by mala byť na priateľnej hranici.

S týmito všetkými poznatkami bol teda vypracovaný akýsi model ľudského vnímania hudby.[30] Signál sme rozdelili na menšie segmenty podľa obsahu, ktoré obsahovali. Rytmus sme reprezentovali pomocou krivky hlasitosti, zafarbenie tónov pomocou spektrogramu s 25 pre ľudské ucho kritickými kanálmi a výšku tónov pomocou 12 dimenzionálneho vektoru zvaného chroma. Dokopy nám teda jeden segment opisuje 12 (chroma) + 25 (timbre) + 1 (krivka hlasitosti) znakov. V predchádzajúcich metódach sa využívala len informácie o priemere hlasitosti. Z empirických skúseností sa však zistilo, že viac je využiteľná informácia o maxime z tejto krivky a tak v našej metóde pridáme 5 dynamických znakov, ktoré budú vychádzať z hlasitostnej krivky. Hlasitosť pri nástupe, maximálna hlasitosť, hlasitosť pri odsadení beatu, dĺžka segmentu a čas, v ktorom dosiahla krivka maximálnu hlasitosť s ohľadom na čas nástupu ( $t_{maxRel} = t_{max} - t_{onset}$ ). Ked' teda prirátame k našim predchádzajúcim 37 znakom popisujúcich segment (timbre =

25, chroma = 12) našich nových 5 znakov vychádzajúcich z hlasitostnej krivky, vychádza nám, že každý segment nám popisuje 42 znakov.



obrázok 5-6 Reprezentácia skladby s jedným nástrojom (klavír). Zhora dole: Zvuková vlna, hlasitostná funkcia, spektrogram, chromogram, chromogram pre segment. [30]

# 6. Implementácia

V nasledovnej časti poukážem na moju implementáciu, ktorá sa snažila prezentovať výhody a nevýhody najpoužívanejších techník na vizualizáciu hudby ako sú FFT a Beat Detection algoritmy. Ďalej ukážem príklad ako by mohol vyzeráť vizualizačný systém, ktorý by obsahoval skutočné údaje o hudbe, ktorú sa snaží vizualizovať. Tiež v tejto kapitole budem prezentovať moju predstavu viacerým ľuďom a diskutovať s nimi o tom, či vôbec taký prístup má pre nich zmysel a využitie. Pri implementácii som použil experimentálny jazyk Processing a spolupracoval som so spoločnosťou EchoNest a ich produkтом Musical Brain, ktorú založili Tristan Jehan a Brian Whitman z MIT Media Labs.

## 6. 1 FFT

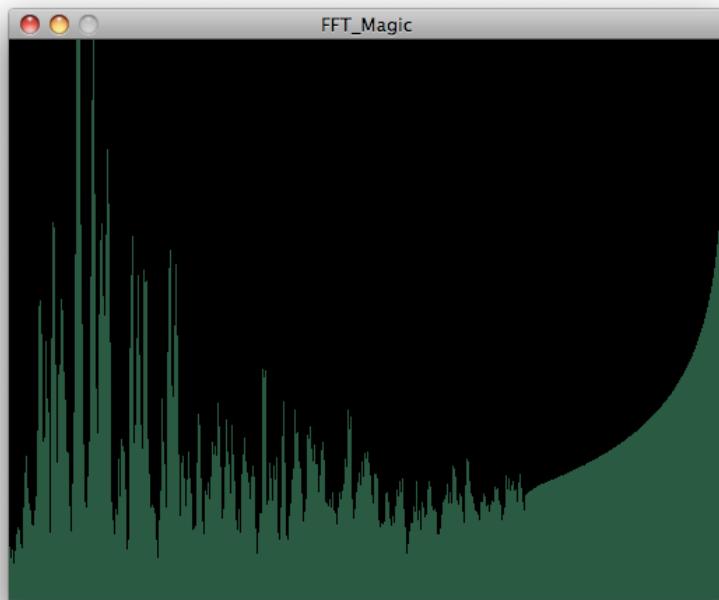
Mojou prvou úlohou bolo naimplementovať asi najrozšírenejšiu techniku na spracovanie zvukového signálu do dátovej podoby. FFT som implementoval v podobe aplikácie FFT Magic, ktorá zoberie načítanú skladbu a aplikuje spomínanú frekvenčnú analýzu v podobe ako je vysvetlená v kapitole 2. 2. 2. Tento algoritmus na vstupe dostane dva parametre, jeden je veľkosť buffera a druhý tzv. sampleRate. Priamo v implementácii zadám len veľkosť buffera, pretože máme zadefinovaný aj konštruktor s jedným parametrom, ktorý premennej sampleRate dá preddefinovanú hodnotu 44100. Na vysvetlenie týchto parametrov uvediem príklad. Skonštruujem FFT objekt len s jedným parametrom 1024, to znamená, že do sampleRate sa automaticky priradí hodnota 44100 Hz. V takomto prípade (takýto istý prípad je použitý v aplikácii FFT Magic) bude výsledné spektrum obsahovať frekvencie pod 22010 Hz, čo je vlastne polovica *sampleRate*. Táto frekvencia sa nazýva aj Nyquistova podľa Henryho Nyquista [36]. Ako vyplýva z teórie v kapitole 2. 2. 2 spektrum neobsahuje individuálne frekvencie, ale takzvané frekvenčné pásma, ktoré sú centrované okolo príslušných frekvencií. Ak chcem teda zistiť na akej frekvencii je centrované napríklad 10 frekvenčné pásma stačí si to vyrátať nasledovným postupom:  $10/1024 * 44100 = 0.009765625 * 44100 = 430.6640625$ . Z toho teda vyplýva 10te frekvenčné pásma je centrované na frekvencii 431 Hz. Celkový počet frekvenčných pásiem sa sice väčšinou rovná

celej dĺžke signálu (44100) ale prístup je dostupný len k frekvenciám nižším ako Nyquistova frekvencia. Inak povedané, ak máme signál dĺžky  $L$ , vždy budeme mať k dispozícii  $L/2$  frekvenčných pásiem.

Takýmto spôsobom sme teda získali dátu, ktoré sa v ďalšom procese vizualizujú. Pri tomto prístupe sa v podstate najväčším problémom stáva fantázia a predstavivosť samotného tvorca. K dispozícii máme rapídne sa meniace premenné a je to len na tvorcovej predstavivosti ako si s nimi poradí. V aplikácii FFT Magic môže užívateľ jednotlivými skratkami prepínať medzi 4 vizuálnymi módmi. V prvom z nich sú jednoducho vykreslené všetky frekvenčné pásma pomocou obdĺžnikov, v druhom sa vykresľuje elipsa ktorej sa menia parametre na základe údajov z FFT a užívateľ môže meniť farbu pohybom myše, v tretej sa využívajú kružnicové oblúky z kružnice a v štvrtom som použil časticový systém (častice sa vytvoria a padajú smerom dole). Užívateľ tiež môže zapínať a vypínať prekreslovanie pozadia, čo mu umožňuje vytvárať ďalšie efekty. Medzi všetkými vizualizačnými módmi môže užívateľ počas vizualizácie prepínať v reálnom čase.

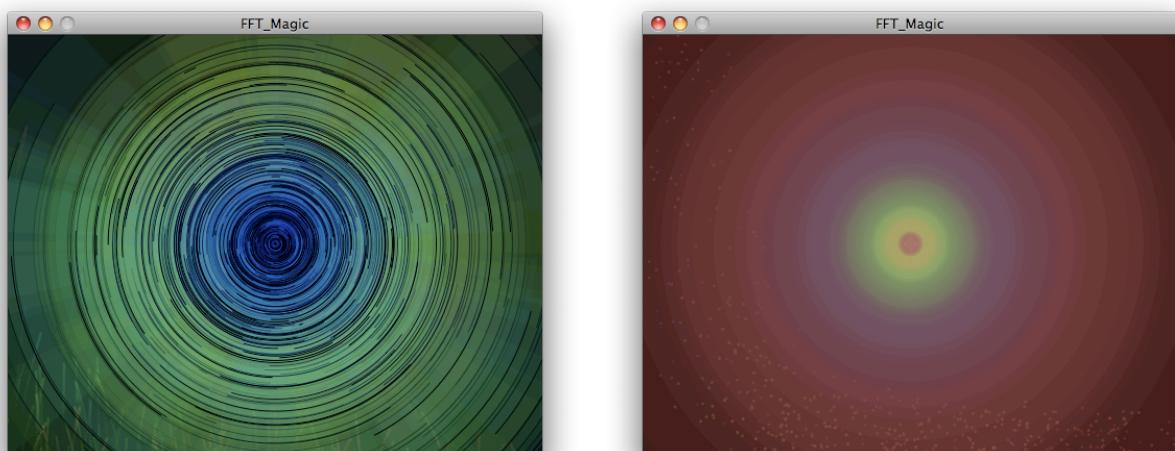
V mojom prípade sa samozrejme nejedná a veľmi vizuálne príťažlivé vizualizácie. Ale v mojom prípade, nebola ich atraktívnosť ani primárnu požiadavku, hlavnou úlohou tejto aplikácie bolo prezentovať techniku fourierovej transformácie a jej použitie pri vizualizačných technikách. Ako sa dá vidieť aj pri mojich vizualizáciach, aj pri veľmi základných vykresleniach vieme vďaka variabilným premenným dostať celkom zaujímavé výsledky. Výhodou tejto techniky je to, že jednoducho získame veľké množstvo dát, ktoré nám dajú absolútну voľnosť pri vizualizácii. Graficky nadanému človeku by tieto dátu mohli úplne stačiť na vytvorenie zaujímavej a atraktívnej vizualizácie. Ďalšou výhodou, ktorú má táto technika, je že veľmi ľahko vieme získať a vidieť celé zvukové spektrum (ako na obrázku 6-1). To môžu využiť rôzni zvukoví inžinieri pri dodačovaní zvuku v štúdiách. Často je totiž zvuk ovplyvnený aj charakterom monitorov a reproduktorov (najdrahšie štúdiové monitory sú charakteristické práve tým, že nemajú žiadny charakter a púšťajú zvuk presne taký aký je). Vďaka takýmto dátam potom inžinieri vidia, ktoré frekvencie sú príliš potichu a môže sa stať, že ich niektorí poslucháči nebudú počuť. Skladbu, ktorú vidíte zvizualizovanú

na obrázku 6-1 som nahrával v svojom byte cez lacný mikrofón a nijako som ju ďalej neupravoval. Z analýzy sa dá zistíť, že vysoké frekvencie nemajú skoro žiadne energie a preto sa pri procese normalizácií veľmi umelo zvýšili (v pravej strane na obrázku).



obrázok 6-1 Zvukové spektrum

Najväčšou nevýhodou tejto techniky je to, že v skutočnosti nemá s hudbou skoro nič spoločné. Jediné to, že sa sústredíuje na energie zvuku, ktorým sa hudba vytvára.



obrázok 6-2 Rôzne módy vizualizácie aplikácie FFT Magic. Naľavo použité kružnicové oblúky a vypnutím prekresľovania pozadia a pravo jednoduché vykreslovanie elips meniacie parametre podľa dát z Fourierovej transformácie

## 6. 2 Beat Detection

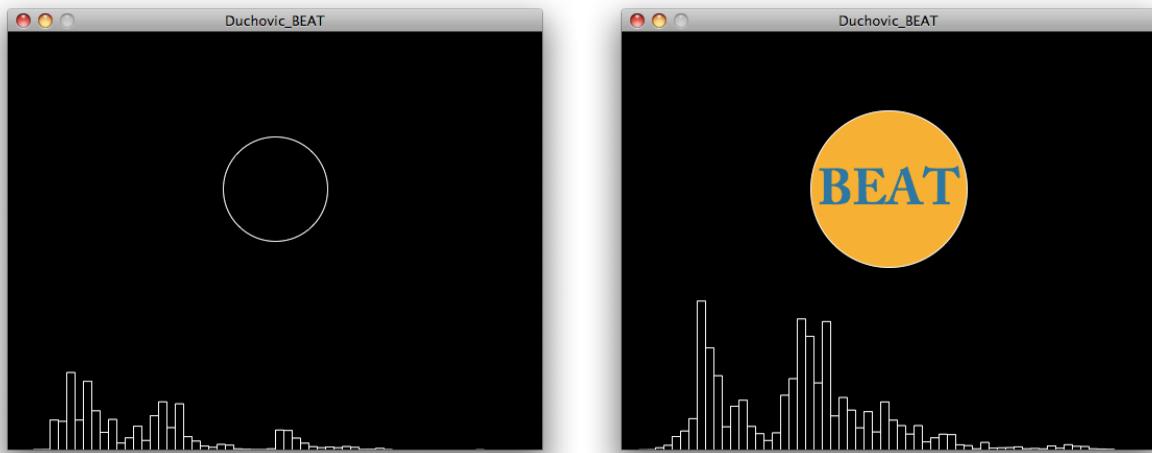
V druhej časti mojej implementácie som mal za úlohu implementovať a prezentovať výhody a nevýhody Beat Detection techniky. Výstupom beat detection techník by mal byť istý *beat*, ktorý je charakteristický pre danú skladbu. Definícia beatu podľa slovníka [37] je, že označuje vizuálne, zvukové alebo mentálne označenie metrických údajov v hudbe. Inými slovami a najjednoduchšie povedané, jeden z najzákladnejších beatov každý z nás prežíva keď si klopká nohou. Preto sa často označuje problém detektie beatov aj ako “foot-tapping” (nohou poklepávajúci) alebo “clapping” (tlieskajúci) problém.

Existuje veľké množstvo metód na daný problém, najvýznamnejšie som spomenul v kapitole 3. Ja som sa rozhodol naimplementovať modifikovanú verziu algoritmu uvedeného v kapitole 3. 1 od autora Frederica Patina [26]. Keďže tento algoritmus sa nezaobíde bez frekvenčnej analýzy (ako v podstate žiadne beat detection algoritmus), súčasťou aplikácie je aj výpočet zvukového spektra, ktoré som spriemeroval do 64och frekvenčných pásiem. Ako hovorí teória v kapitole 3. 1 beat detektujeme vtedy, ak zachytíme energiu, ktorá je nadpriemerne zväčšená oproti lokálnemu priemeru.

Základom tejto aplikácie je trieda `BytDetekuj`, ktorá obsluhuje celý proces detektovania beatu. V konštruktore má objekt dva parametre, prvý hovorí o počte samplov, ktoré budú reprezentovať instantnú energiu a druhý o počte samplov, reprezentujúcich históriu. Dôležitou premennou je teda `history_buffer`, ktorý je reprezentovaný ako  $(r/s)$  - prvkové pole floatov ( $r$  - počet samplov definujúcich instantnú energiu,  $s$  - počet samplov definujúcich history buffer), pričom v každom prvku pola je energia daných  $r$  samplov. V našom konkrétnom prípade je  $r = 44100$  a  $s = 1024$ , to znamená, že náš `history_buffer` je 43 prvkové pole, pričom v `history_buffer[0]` je energia najnovších samplov a v `history_buffer[42]` vždy energia najstarších. Na našom `history_buffery` vyrátame každých  $r$  samplov priemernú energiu, ktorú porovnáme s instantnou energiou. Po vyrátaní priemernej lokálnej energie, najstaršie sample z `history bufferu` zmažeme, indexy posunieme a prichystáme tak miesto pre energiu z najnovších samplov. Následne máme teda

dve premenné *instant* (instantná energia) a *E* (priemerná lokálna energia) Beat bude detekovaný vďaka jednoduchému porovnaniu - ak bude premenná *instant* väčšia ako *C*-násobok premennej *E*. Tým sa dostávame k dôležitosti výberu konštanty *C*. Jej výber záleží aj od typu hudby, pri hudbe, kde sú beaty veľmi intenzívne, ako napríklad (techno, hip-hop) by mala byť táto konštanta celkom vysoká (okolo 1.4) naopak pri hudbe, ktorá obsahuje veľa hluku ako napríklad rock by mala byť nižšia (okolo 1.2). V našej implementácii sme implementovali dynamickú konštantu, ktorá sa nemusí zadať pri vytvorení objektu, ale sama sa mení podľa zmien energií. Máme teda premennú *V* (variance = zmena), ktorá obsahuje priemer rozdielu energií v *history\_bufferi* s lokálnym priemerom (*history\_buffer[i]* - *E*, pre *i* = 0,1...42). Tým získame priemernú zmenu energie v premennej *history\_buffer* na základe ktorej vieme dynamicky meniť konštantu *C*. Čím väčšia je premenná *V*, tým menšia by mala byť konštanta *C*, pretože to znamená, že zmeny energií sú veľmi vysoké a časté, takže by algoritmus mal byť senzitívnejší. Naopak ak nadobúda *V* malú hodnotu, znamená to, že konštante *C* môžme priradiť väčšiu hodnotu, pretože energie sa menia veľmi málo a keď príde beat, bude to veľmi ľahko pozorovateľné. V našej konkrétnej implementácii sú vzťahy zdefinované rovnako ako v článku od Frederica Patina [26], to znamená, že ak je *V* = 200 tak *C* = 1.0 a ak *V* = 25 tak *C* = 1.45. Z týchto vzťahov získame matematický vzťah  $C = (-0.0025714 * V) + 1.5142857$ , ktorý je v implementácii priamo použitý.

Z hľadiska vizualizácie som túto metódu poňal veľmi jednoducho. V spodnej časti aplikácie sa vizualizuje spriemerovaná fourierova transformácia v podobe obyčajných obdĺžnikov a ak vráti metóda *jeNastup* hodnotu *true* zaznamená sa beat - vizuálne sa vykreslí elipsa s nápisom BEAT (obrázok 6-3). Pre túto vizualizáciu som nahral špeciálne audio, na ktorom sa dajú veľmi jednoducho prezentovať výhody aj nevýhody tejto techniky. Skladba má trvanie okolo minúty a začína tichou pasážou len s klávesmi. Pri tomto začiatku môžme vidieť nevýhodu tejto techniky v tom, že keďže pracuje s lokálnym priemerom energií, pri tichých pasážach zaznamenáva beat aj pri jemných výkyvoch energie. Čiastočne tomu pomáha dynamická konštanta *C* ale je to len čiastočná výpomoc. Pri zložitejších



obrázok 6-3 Vizualizácia beat detection algoritmu. Na ľavej strane môžete vidieť normálny stav algoritmu a na pravej stav, keď je rozpoznaný beat.

systémoch ako je opísaný napríklad v kapitole 3.2 má každý beat (v prípade BTS z kapitoly 3.2 každý agent) vlastnú premennú, kde sa zaznamenáva spoľahlivosť prehlásenia, že sa zachytí beat. To v tejto implementácii chýba ale tiež by to problém nevyriešilo úplne, implementácia by sa vtedy prispôsobila tak, že by zaznamenala beat, len ak by spoľahlivosť beatu bola nad istou hranicou (tá by sa zistila empiricky). Tento problém so spoľahlivosťou majú v podstate všetky dostupné algoritmy venujúce sa problému detektovania beatov a ani jedna z momentálne existujúcich techník nie je 100 percentne úspešná na všetky hudobné žánre.

Avšak ako skladba pokračuje, pridajú sa k nej bicie nástroje. Najprv sa pridá basový bubon s rytmičákom v nie až tak pravidelnom rytme, čím sa detekor až tak neuchytí. Akonáhle však prejde basový bubon do pravidelného hrания na každú štvrtovú notu, detektor zachytí s vysokou úspešnosťou každý jeho úder. Práve kvôli tomu som nahral vlastný podklad, aby som v minútovej skladbe poukázal na rôzne situácie, ktoré v rôznych žánroch môžu nastať. Najpresnejší sa algoritmus stal, ak v skladbe nastúpil pravidelný a úderný beat, najčastejšie sa takýto beat vyskytuje v techno žánri, ale ak bol beat rozloženejší a zložitejší úspešnosť už taká vysoká nebola. Riešením je implementovať podobný systém ako Masataka Goto v kapitole 3.2, kde využíva systém agentov, ktorí pracujú už s dopredu definovanými vzormi najčastejších bubenových vzorov (viac v kapitole 3.2)

## 6. 3 Analýza hudby

V tretej aplikácii, nazvanej Music\_analysis stála predomnou neľahká úloha naplniť moju predstavu o analýze skutočnej hudby. Ako jedno z prvých riešení mi prišla na um analýza hudby z priateľného zdroja - MIDI. Z tohto zdroja by človek nemal totiž problém získať akékoľvek dátu a následne ich vizualizovať. Avšak keďže MIDI ako zvukový zdroj nie je veľmi rozšírený (asi tiež nepoznáte nikoho, kto doma počúva hudbu v MIDI), nakoniec som danú myšlienku neimplementoval. Každopádne si tento nápad nezaslúži zavrhnutie, už len kvôli jednoduchému prístupu k všetkým hudobným dátam, ktoré MIDI poskytuje. Veľkou výhodou je aj to, že tieto dátu sú vždy korektné a nemusíme riešiť chybovosť a spoločnosť dát.

Súčasne s nápadom vizualizovať MIDI som študoval ako je na tom technológia v extrahovaní hudobných dát zo skutočných zvukových signálov. Po čítaní veľkého množstva špecializovaných článkov som sa dostal k spoločnosti EchoNest a jej produktu Musical Brain. K nej som sa dostal vďaka článkom od autorov Tristana Jehana a Briana Whitmana. Títo autori pôsobili na MIT v sekcií Media Lab a prezentujú efektívne metódy práve na získavanie hudobných dát a následnú resyntetizáciu. Neskôr som sa dostal aj k ich dizertačným prácам, na základe ktorých založili práve spomínanú spoločnosť EchoNest. Ich hlavný produkt nesie názov Musical Brain a je to jeden z najkvalitnejších systémov svojho druhu. Jeho primárne využitie bolo na vyhľadávanie a doporučovanie hudby na internete. Tento systém pracuje v troch hlavných fázach, v prvej analyzuje milióny blogov, recenzií, diskusných fór a snaží sa pochopiť dátu, ktoré opisujú daného autora, album alebo skladbu, v ďalšej sleduje vývoj hudobných trendov pomocou sledovania sťahovaní skladieb, ich streamovaní publikovaní atď. V poslednej fáze analyzuje skladby z hudobného hľadiska pomocou metód opísaných v kapitole 5. Práve táto posledná fáza ma donútila začať sa zaoberať týmto systémom aj naďalej. Koncom roku 2008 otvorila táto firma na webe možnosť zaregistrovať sa ako developer a získať svoj vlastný klúč, pomocou ktorého sa dá pristupovať k systému Musical Brain. Aj keď som teda nedostal priamo prístup k všetkým metódam a zdrojovému kódu (čo je samozrejmé, keďže táto spoločnosť patrí naozaj k najúspešnejšej v danej oblasti), už z výsledkov som videl veľkú

podobnosť (až totožnosť) s metódami popísaných v článkoch a dizertačných prácach od Tristana Jehana a Briana Whitmana. Nevýhoda takejto spolupráce je nutnosť byť pri analýze a následnej vizualizácie online, pretože požadovaná skladba sa musí najprv uploadnúť na server, kde prebehne pomocou metód opísaných v kapitole 5 analýza, ktorá sa developerovi vráti vo forme xml dát.

Mojou implementáciou je teda trieda DuchEchoNest, ktorá sa volá s dvoma hlavnými parametrami. Prvým je klúč, pomocou ktorého získavame prístup k systému Musical Brain (získame ho po zaregistrovaní sa ako developer spoločnosti EchoNest). Druhým je string, ktorý popisuje cestu k súboru, ktorý budeme uploadovať na server. Na spracovanie dát som použil xml wrapper od Kamela Mehlkoufa z jeho echonest knižnice, ktorá mi nevyhovovala pre niektoré obmedzenia (ako napríklad nemožnosť využitia súboru, ktorý neobsahoval metadata atď). Tento xml funguje tak, že z výstupu vo forme xml zadefinuje svoje vlastné triedy (ako napríklad ENKey, ENTImeSignature atď), ktoré majú rovnaké premenné ako ich xml ekvivalenty. Napríklad z premennej *key*, ktorá ma v xml atribúte *confidence* vznikne trieda ENKey s premenou confidence. Tú neskôr spracuje moja vlastná trieda v podobe premennej *tonina*, ktorú spracujem podľa referencií EchoNestu. Zaujímavé je tiež spracovanie segmentov (krátká časť zvuku, ktorá je harmonicky alebo farebne uniformná), z ktorých budeme vytvárať pole segmentov, ktoré obsahujú premennú *pitches*, z ktorej vyextrahujem dátu pre vznik 12 dimenzionálneho vektora *chroma* (kapitola 5), z ktorého vieme získať základné harmonické vzťahy pre daný segment. Na obrázku 6-4 môžete vidieť dátu, ktoré sa mi podarilo získať v spojení mojej triedy so systémom Musical Brain od spoločnosti EchoNest. Pri každom type získaných dát je prístupná premenná, ktorá hovorí o spoľahlivosti. Ako vidíte z obrázku, pre každý segment máme zadefinované premenné, ktoré hovoria o začiatku segmentu a o jeho trvaní. Začiatok a trvanie sú zadefinované takisto pre každú sekciu (vypísaná je len premenná počet sekcií), to znamená, že vieme získať tóny znejúce na začiatku a na konci sekcie. Jednoducho zistíme čas začatia alebo konca (čas začatia + trvania sekcie) sekcie, a nájdeme segment, ktorý v danom čase začína. Keď už vieme segment, nie je problém nájsť jeho harmonickú reprezentáciu (chromu).. V mojej triede som ešte napríklad dodefinoval metódu *reprezentant\_segmentu*, ktorá

vráti tri najvýraznejšie tóny v danom segmente. Presnejšie povedané vráti 12 prvkové pole podobné premennej *chroma*, s rozdielom, že naplnené budú len indexy, ktoré zodpovedajú tónom s maximálnymi hodnotami. Dáta získane touto metódou môže byť zaujímavé vizualizovať, pretože v skutočnosti to nemusia byť tóny, ktoré sú dôležité z hľadiska harmónie, ale tie ktoré najviac počuť (bude vidieť rozdiel medzi harmonickým zámerom a skutočnosťou). Tiež som zdefinoval metódu, ktorá podľa tóniny a charakteru skladby vráti harmonicky akceptovateľné tóny. Takto získané dátá vedia vysvetlovať dôvod informácií o hudobnom obsahu a vizualizovať ich možno podľa vlastnej ľubovôle.



obrázok 6 4 Dáta získané pomocou triedy DuchEchoNest a systému Musical Brain. Na server bola uploadnutá skladba od Amy Winehouse: Love is A Losing Game.

Možnosti sú rôzne, dajú sa napríklad vizualizovať už spomínané tri najvýznamnejšie tóny v každom segmente, pri nástupe na novú sekciu môžme ukázať súzvuk tónov, ktoré sú v danom čase najsilnejšie, počas skladby môžme zakomponovať aj textové vizualizácie, ktoré môžu odovzdávať informácie o tempe, tónoch, ktoré je možné zahrať do skladby tak, aby boli v harmónii a mnohé iné dátá získané z našej triedy DuchEchoNest. .

## 6. 4 Konfrontácia výsledkov

Výsledky mojej práce môžte nájsť na internete na adrese <http://duchovic.igo.sk>. K dispozícii tam nájdete 3 aplikácie spomínané v implementácii, ktoré som publikoval v podobe appletov. Prvá aplikácia FFT Magic prezentuje techniku fourierovej transformácie, druhá Duchov\_Beat algoritmus na riešenie problému detekcie beatu spolu s jeho výhodami a nevýhodami a tretia Music\_analysis implementuje triedu DuchEchoNest na obsluhu systému Musical Brain od firmy EchoNest na analýzu hudobného obsahu (kapitola 5).

Tieto výsledky som prezentoval osobne dvom typom ľudí. Jedny z nich sú profesionálny hudobníci, ktorí sú oboznámení s hudobnou teóriou, harmóniou a rytmickými pravidlami a druhou skupinou sú každodenní, hudobne nevzdelaní ľudia.

### 6. 4. 1 Profesionálny hudobníci

Všetci hudobníci, ktorým som prezentoval výsledky boli oboznámení s tématikou, pretože sa už zúčastnili na prieskume v kapitole 4. Prezentáciu som začal vysvetlením princípu fungovania dnešných vizualizérov (väčšina z nich ich nevyužívala, alebo využívala minimálne) a ukázal im prvé dve aplikácie prezentujúce techniku fourierovej transformácie a detekcie beatov. Po prezentovaní všetkých výhod a nevýhod sa 100 percent všetkých hudobníkov zhodlo na tom, že tieto metódy vôbec nezodpovedajú spôsobom akým oni vnímajú hudbu. Tiež som im prezentoval, niektoré profesionálne vizualizéry, ktoré s týmito technikami pracujú, a aj keď' uznali, že vizuálne sú veľmi príťažlivé, s hudbou, ktorú vizualizujú toho veľa spoločného nemajú.

Po tomto prehlásení sa začalo diskutovať o inom prístupe a niektorí z nich boli celkom nadšení z predstavy, že by priamo z audio skladby vedeli získať informácie o hudobnom obsahu. Spomínať sa napríklad využitie pri potrebe naštudovať si veľké množstvo skladieb za krátky čas, stačilo by si pustiť vizualizáciu a hudobník by nemusel tráviť čas nad sťahovaním konkrétnych tónov.

Vizualizácia by mu sama vedela vypovedať v akej tónine a aké tóny v danom čase znejú. Bubeník zasa uprednostňoval informáciu o bpm, aj keď v zápäti dodal, že to je informácia, ktorú vie sám získať za veľmi krátkej čas.

Po tejto diskusii som im ukázal vizualizáciu dát získaných z aplikácie Music\_Analysis a zhodli sa, že aj keď sa vizualizujú už dátá súvisiace priamo s hudobným obsahom, stále by ich to neprinútilo využívať vizualizér na každodennej báze. Tiež však bolo povedané, že by im viac vyhovovalo keby daná vizualizácia bola viac orientovaná len na výpis informácií, ktoré by vedeli využiť. Jedinou výnimkou bol skladateľ hudby, ktorý keďže pracuje veľa s dokonalou vizualizáciou hudby (noty) bol zvedavý, ako by vyzerala vizualizácia, ktorá by presne vizualizovala tóny, ktoré znejú a harmonické vzťahy medzi nimi. Následne som mu vysvetlil možnú chybovosť údajov pri spracovaní takýchto údajov priamo z audio signálu a že vizualizácia takéhoto typu je možná len z MIDI zdroja (zaujímave je, že jeho nadšenie neopadlo a presvedčal ma, že takú vizualizáciu by využíval, aj keby pracovala len s MIDI).

## 6. 4. 2 Hudobní analfabeti

Prezentácia výsledkov začínala podobne ako v prvom prípade. Vysvetlil som spôsoby, akým vzniká vizualizácia a prezentoval prvé dve aplikácie, na základe ktorých usúdili, že takéto dátá nie sú dostatočné. Avšak pri prezentácii profesionálneho, vizuálne atraktívneho vizualizéra väčšine nevadilo jeho nezodpovedanie s hudbou a vyzdvihovali jeho vizuálne spracovanie.

Následne sme sa dostali k ďalšej časti mojej implementácií, kde som ukázal vizualizáciu dát, ktoré súvisia s hudobným obsahom. Väčšina opýtaných odpovedali, že bližšie k hudbe sa im zdalo to predchádzajúce zobrazenie, ktoré sa aspoň zväčšilo, keď nastala energetická zmena. Z rozhovorov teda môžem povedať, že väčšina hudobne nevzdelených ľudí vníma hudbu skôr ako energiu, to znamená, že prístup prvých dvoch aplikácií, im je bližší a vedia sa s nim lepšie stotožniť.

Zaujímava situácia však nastala pri ľuďoch, ktorí hrajú na nejaký hudobný nástroj, ale ovládajú ho len na začiatočníckej úrovni. Takíto respondenti neskrývali nadšenie z prístupu, ktorý by im doporučil tóny, ktoré môžu hrať do svojich oblúbených skladieb a nebyť mimo harmónie. Tiež sa im páčila predstava, že by mohli zistiť akordy svojich oblúbených pesničiek, bez zbytočného prehľadávania internetu, prípadne stiahovaním tónov podľa sluchu (začiatočníkovi to môže trvať celé hodiny). Samozrejme, aj im som objasnil, že takáto analýza nemusí byť stopercentne korektná, avšak ani táto informácia ich neobrala o nadšenie.

# Záver

Vizualizačné programy hudby sú v dnešnej dobe vo veľmi zvláštnom stave. Ako ukázal aj prieskum v kapitole 4, v podstate neexistujú ľudia, ktorí by vizualizér hudby využívali každodenne. Doba rozkvitu vizualizérov bola za čias, keď počítač nedisponoval toľkými graficky prítulnými aplikáciami a keď každý plynulý a vizuálne aspoň trochu atraktívny pohyb na obrazovke spôsoboval údiv. Tieto časy sú však dávno preč a zaujať užívateľa v dnešnej dobe nie je až také jednoduché.

Môj prístup, ktorým som chcel zmeniť zaužívaný systém, ktorým vizualizéry fungujú, bolo zmeniť dátu, ktoré sa vizualizujú. Mojm prvotným cieľom boli ľudia, ktorí sa hudbe venujú profesionálne a mohli by danú vizualizáciu využívať denne pri svojich hudobných činnostiach. Po diskusiách a prezentáciach mojich nápadov som došiel k názoru, že samotná vizualizácia by pre nich aj tak zaujímavá nebola, išlo by im hlavne o dátu o hudobnom obsahu.

Zaujímavým spôsobom však vyšla na povrch úplne nová cielová skupina - a to sú práve začínajúci muzikanti. Pre nich by mohol byť takýto druh vizualizácie zaujímavý, pretože by sa pri počúvaní svojich oblúbených skladieb mohli naučiť aj ako ich hrať. Tiež by podobná vizualizácia mohla pomôcť pri učení improvizácie, užívateľ by si vybral podklad, do ktorého by chcel improvizovať a vizualizácia by mu vďaka svojim informáciám pomáhala s výberom množiny tónov, ktoré by mohol hrať.

Ďalšie využitie by mohlo byť zakomponovanie hudobných dát do už používaných dát na vizualizáciu. Keďže sú takéto dátá oveľa bližšie k skutočnej hudbe, graficky nadaní tvorcovia by s nimi vedeli vytvoriť určite veľmi príťažlivé vizualizácie. Po istej optimalizácii by teda takéto vizualizácie nemuseli slúžiť len ako podprahový doplnok k hudbe ale aj ako študijna pomôcka a taktiež ako pomocník pri potrebe naučiť sa veľké množstvo repertoáru za krátky čas.

## Zdroje

- [1] Dictionary.com [online]. 2004 [cit. 2009-08-21]. The American Heritage® Dictionary of the English Language.  
Dostupné z WWW: <<http://dictionary.reference.com/browse/music>>
- [2] Dictionary.com [online]. 2009 [cit. 2009-08-21]. Unabridged. Random House Inc. Dostupné z WWW:  
<<http://dictionary.reference.com/browse/music>>.
- [3] HAWES, Neil. Neilhawes.com [online]. 2003 [cit. 2009-10-27]. History of notation. Dostupné z WWW:  
<<http://neilhawes.com/sstheory/theory22.htm>>.
- [4] Psychedelia (light synthesizer) In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-04-20]. Dostupné z WWW:  
<[http://en.wikipedia.org/wiki/Psychedelia\\_\(light\\_synthesizer\)](http://en.wikipedia.org/wiki/Psychedelia_(light_synthesizer))>.
- [5] Music visualization In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-04-20].  
Dostupné z WWW: <[http://en.wikipedia.org/wiki/Music\\_visualization](http://en.wikipedia.org/wiki/Music_visualization)>.
- [6] AYERS, Larry. Linux.org.uk [online]. 1997 [cit. 2009-10-28]. Visual music: The Linux Port of Cthuga. Dostupné z WWW:  
<<ftp://ftp.linux.org.uk/LDP/LDP/LG/issue17/cthuga.html>>.
- [7] AbsoluteAstronomy.xom [online]. 2009 [cit. 2009-10-28]. MilkDrop.  
Dostupné z WWW:  
<<http://www.absoluteastronomy.com/topics/MilkDrop>>.
- [8] GEISS, Ryan. Nullsoft.com [online]. 2007 [cit. 2010-04-20]. MilkDrop.  
Dostupné z WWW: <<http://www.nullsoft.com/free/milkdrop/>>.
- [9] O'MEARA, Andrew. 55ware.com [online]. 2006 [cit. 2009-10-30]. Home. Dostupné z WWW: <<http://www.55ware.com/index.html>>,
- [10] Soundspectrum.com [online]. 2009 [cit. 2009-10-30]. G-Force.  
Dostupné z WWW: <<http://www.soundspectrum.com/g-force/>>. ]
- [11] Podcomplex.com [online]. 2009 [cit. 2009-11-4]. Physics of sound.  
Dostupné z WWW:  
<<http://www.podcomplex.com/guide/physics.html>>.

- [12] RUSSEL, Kevin. Umanitoba.ca [online]. 1997 [cit. 2009-11-4]. Sound Waves. Dostupné z WWW:  
<<http://www.umaniitoba.ca/faculties/arts/linguistics/russell/138/sec4/acoust1.htm>>.
- [13] Thinkquest.org [online]. 1999 [cit. 2009-11-4]. Physics of sound 2. Dostupné z WWW:  
<<http://library.thinkquest.org/19537/Physics2.html>>.
- [14] Thinkquest.org [online]. 1999 [cit. 2009-11-4]. Physics of sound 3. Dostupné z WWW:  
<<http://library.thinkquest.org/19537/Physics3.html>>.
- [15] Astro.oma.be [online]. 2002 [cit. 2009-11-12]. Image 052. Dostupné z WWW:  
<[http://www.astro.oma.be/ICET/bim/text/dierks\\_fichiers/image052.gif](http://www.astro.oma.be/ICET/bim/text/dierks_fichiers/image052.gif)>.
- [16] KASTNER, Ryan. Ece.ucsb.edu [online]. [cit. 2009-11-12]. Image 032. Dostupné z WWW:  
<[http://www.ece.ucsb.edu/~kastner/ece15b/project1/fft\\_description\\_files/image032.jpg](http://www.ece.ucsb.edu/~kastner/ece15b/project1/fft_description_files/image032.jpg)>.
- [17] Mathworks.com [online]. [cit. 2009-11-12]. Image. Dostupné z WWW:  
<[http://www.mathworks.com/access/helpdesk/help/toolbox/vipblk/ref/ch\\_block\\_ref2189.gif](http://www.mathworks.com/access/helpdesk/help/toolbox/vipblk/ref/ch_block_ref2189.gif)>
- [18] ZARA, Jiri, et al. Moderni pocitacova grafika. Brno : Computer Press, 2004. 609 s. ISBN 80-251-0454-0str. 43-45
- [19] ZARA, Jiri, et al. Moderni pocitacova grafika. Brno : Computer Press, 2004. 609 s. ISBN 80-251-0454-0str. 574-576
- [20] Masataka Goto and Yoichi Muraoka: Music Understanding At The Beat Level -Real-time Beat Tracking For Audio Signals ---, pp.68-75, August 1995.
- [21] Masataka Goto: An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds, Journal of New Music Research, Vol.30, No.2, pp.159-171, June 2001.
- [22] Masataka Goto: A chorus section detecting method for musical audio signals, ICASSP 2003 (The 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing) Proceedings, pp.V-437-440, April 2003.

- [23] Masataka Goto and Yoichi Muraoka: Beat Tracking based on Multiple agent Architecture - A Real-time Beat Tracking System for Audio Signals, Proceedings of The Second International Conference on Multiagent Systems, pp.103-110, December 1996.
- [24] BANKS, Kevin. Embedded.com [online]. 2002 [cit. 2010-04-20]. Goertzel Algorithm. Dostupné z WWW: <<http://www.embedded.com/story/OEG20020819S0057>>.
- [25] Goertzel algorithm In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, , [cit. 2010-04-20]. Dostupné z WWW: <[http://en.wikipedia.org/wiki/Goertzel\\_algorithm](http://en.wikipedia.org/wiki/Goertzel_algorithm)>
- [26] PATIN, Frederic. Gamedev.net [online]. 2003 [cit. 2010-04-20]. Beat Detection Algorithms. Dostupné z WWW: <<http://www.gamedev.net/reference/articles/article1952.asp>>.
- [27] X. Serra. A system for Sound Analysis/Transformation/Synthesis Based on a Deterministic Plus Stochastic Decomposition. PhD thesis, CCRMA, Department of Music, Stanford University, 1989.
- [28] X. Serra and J. O. Smith. Spectral modeling synthesis: A sound analysis synthesis system based on a deterministic plus stochastic decomposition. Computer Music Journal, 14(4):12–24, 1990
- [29] SCHEIER, Eric D. Music Listening Systems. MIT Media Labs, 2000. 248 s. Dizertační práce. Massachusetts institute of technology.
- [30] JEHAN, Tristan. Creating Music By Listening. MIT Media Labs, 2005. 137 s. Dizertační práce. SCHEIER, Eric D. Music Listening Systems. MIT Media Labs, 2000. 248 s. Dizertační práce. Massachusetts institute of technology.
- [31] JEHAN, Tristan. Event-Synchronous Music Analysis/Synthesis, Proceedings of the 7th International Conference on Digital Audio Effects (DAFx'04). Naples, Italy, October 2004
- [32] B. R. Glasberg and B. C. J. Moore. A model of loudness applicable to time-varying sounds. J. Audio Eng. Soc., 50:331–342, 2002.
- [33] J. Grey. Timbre discrimination in musical patterns. Journal of the Acoustical Society of America, 64:467–472, 1978.
- [34] RINGGENBERG, Kyle; WU, Yi-Chieh. Cnx.org [online]. 2005 [cit. 2010-04-20]. Sinusoidal Harmonic Modeling. Dostupné z WWW: <<http://cnx.org/content/m13206/latest/>>.

- [35] M. A. Bartsch and G. H. Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In Proceedings of IEEE Wokshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pages 15–18, Mohonk, NY, October 2001.
- [36] Wolfram.com [online]. 2010 [cit. 2010-04-20]. Nyquist Frequency . Dostupné z WWW:  
[<http://mathworld.wolfram.com/NyquistFrequency.html>](http://mathworld.wolfram.com/NyquistFrequency.html).
- [37] Dictionary.com [online]. 2009 [cit. 2010-04-20]. Unabridged. Random House Inc. Dostupné z WWW:  
[<http://dictionary.reference.com/browse/beat>](http://dictionary.reference.com/browse/beat).