Automatska transkripcija glazbe

Sekereš, Ivan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

Automatska transkripcija glazbe

Završni rad

Ivan Sekereš

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 07.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Ivan Sekereš
Studij, smjer:	Programsko inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4567, 28.07.2020.
OIB Pristupnika:	19384960523
Mentor:	doc. dr. sc. Hrvoje Leventić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Automatska transkripcija glazbe
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak završnog rad:	Istražiti i opisati postojeće metode i aplikacije za automatsku transkripciju glazbe. Opisati teorijske osnove potrebne za transkripciju glazbe iz audio datoteke. Za praktični dio implementirati algoritam za transkripciju glazbe iz audio datoteke. Rezervirano za: Ivan Sekereš
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	07.09.2023.
Datum potvrde ocjene od strane Odbora:	24.09.2023.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 25.09.2023.

Ime i prezime studenta:	Ivan Sekereš
Studij:	Programsko inženjerstvo
Mat. br. studenta, godina upisa:	R4567, 28.07.2020.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Automatska transkripcija glazbe**

izrađen pod vodstvom mentora doc. dr. sc. Hrvoje Leventić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Područje automatske transkripcije glazbe (engl. *automatic music transcription, AMT*) predstavlja problem kojem je cilj doći do ljudski čitljivih oblika glazbe (kao što su note) računalnom obradom zvučnog zapisa odnosno signala. U ovom radu obuhvaćene su teme koje opisuju glavne značajke glazbe potrebne proces *AMT*-a. Opisane su razine samog područja te objašnjeni osnovni načini izvlačenja značajki i njihovog prikaza. Također su opisane glazbene strukture i dana je implementacija algoritma za detekciju akorda iz glazbe. Opisane su napredne metode i prikazani su primjeri njihovih implementacija te su opisani budući ciljevi i problemi koji su vezani uz područje *AMT*-a. Ovaj rad je strukturiran na način da je u drugom poglavlju sažeto opisana povijest problema *AMT*-a, objašnjene su osnovne značajke glazbe, prikazani formati zapisa glazbe i navedene razine automatske transkripcije. U trećem poglavlju opisane su i uspoređene glavne metode za izvlačenje korisnih značajki iz glazbe, objašnjene su više glazbene strukture i opisan je i implementiran algoritam za detekciju akorda kao praktični dio ovog rada. U četvrtom poglavlju opisani su i dani primjeri kao izvući značajke više (dublje) razine glazbe. U petom poglavlju opisani su problemi i načini naprednijih metoda za proces *AMT*-a te su također dani primjeri u obliku izlaza već postojećih rješenja. U šestom poglavlju navedeni su glavni budući ciljevi i problemi koji postoje u području *AMT*-a.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je opis postojećih metoda i aplikacija za automatsku transkripciju glazbe. Opis teorijskih osnova potrebnih za transkripciju glazbe na računalu. Praktičnim dijelom implementirati korake obrade audio signala, korake izvlačenja značajki glazbe iz obrađenog signala i pokazati naprednije metode za automatsku transkripciju glazbe.

2. Metode za automatsku transkripciju glazbe

2.1. Povijest automatske transkripcije glazbe i trenutačni uspjeh

Kroz povijest ljudi su uvijek bili povezani uz stvaranje i reproduciranje melodije odnosno glazbe. Kako bi se glazbeni uradci sačuvali morali su postojati glazbenici koji su ih prenosili usmenim (slušnim) načinom kao i pismenim. U svrhu zapisa glazbe osmišljene su note odnosno način zapisa melodije na papir odnosno na fizički medij. Kako bi se glazba pravilno zapisala potrebno je znanje i dobar sluh jednog ili više glazbenika, što problem zapisa glazbe otežava. Rijetki su ljudi koji mogu stvoriti točnu transkripciju te je razvojem tehnologije nastala ideja da se proces transkripcije počne izvoditi računalnim putem. Problem odnosno nastanak ideje automatske transkripcije glazbe nastao je ranih 1970.-ih godina. Kako je većina glazbe polifonijska (više instrumenata i/ili glasova), u počecima se promatrao samo problem monofonijske glazbe (jedan instrument ili glas). Prvo kompletno rješenje transkripcije monofonijske glazbe nastalo je 1986. u znanstvenom radu dr.sc. Martina Piszczalskija [1, str. 6]. Piszczalski je u svom radu dao glavne osnove za automatsku transkripciju glazbe na kojima se i dan danas temelji svaki takav sustav. Rad opisuje napredak i razvoj sustava započetog s razvijanjem 1977. godine. U radu je opisana glavna podjela koraka *AMT*-a, a to su sljedeći koraci: analiza i procesiranje signala (određivanje amplituda i početaka noti), formuliranje hipoteza za note u zadanim intervalima i analiziranje konteksta i strukture dobivenih nota kako bi se dobio završni izlaz odnosno notni zapis. James Moorer je 1977. prvi izdao rad u kojem opisuje rad sustava za transkripciju polifonijske glazbe, točnije sustava sposobnog za transkripciju dva instrumenta [1, str. 6-7]. Sustav je imao velika ograničenja, kao što su npr. zvuk instrumenta morao je biti konstantan i precizan te se visina tona i harmonici nisu smjeli preklapati. Michael Hawley je 1993. godine objavio sustav koji je bio u mogućnosti napraviti podosta točnu transkripciju klavira, baziranu na diferencijalnim spektrima dobivenih brzom Fourierovom transformacijom [2, str. 1]. Grupa znanstvenika na Sveučilištu u Tokyu je 1993. stvorila sustav za transkripciju koji je implementirao nove tehnike kao što su Bayesova mreža vjerojatnosti [3, str. 4-5]. Sustav je dodatno unaprijeđen 1995. godine. Znanstvenici u MIT-u predvođeni Keith Martinom napravili su 1996. sustav na blackboard arhitekturi koji je imao mogućnost transkripcije 4 različita klavira odjednom [3, str. 4-5]. Razvojem strojnog učenja i neuronskih mreža te komercijaliziranjem aplikacija za *AMT* obilježeno je 21. stoljeće. Od 2005. godine održava se MIREX (engl. *Music Information Retrieval Evaluation eXchange*), program koji prikuplja sustave za izdvajanje informacija iz glazbe (engl. *Music Information Retrieval, MIR*) te ih evaluira prema zadanim smjernicama [4]. Danas postoji podosta širok izbor aplikacija

odnosno softvera za *AMT* kao što su: *AudioScore, Melodyne, AnthemScore, Transcribe!* i *ScoreCloud*. Takav softver karakterizira velika točnost u području transkripcije monofonijske glazbe i nešto manja, no također podosta velika točnost u području polifonijske transkripcije. Znanstvenici i razne kompanije rade na razvijanju *AMT* sustava odnosno modela baziranih na strojnom učenju i umjetnoj inteligenciji.

2.2. Potrebne značajke glazbe za automatsku transkripciju

Značenje riječi *audio* odnosi se na prijenos, primanje i slanje zvuka koji je u ljudskom slušnom rasponu. Audio signal predstavlja zvuk te je drugačiji od već spomenutih simboličkih reprezentacija jer obuhvaća sveukupni temporalni dojam, dinamiku i tonske mikrodevijacije. Problem kod audio reprezentacije glazbe je taj što frekvencije i visine tonova, trajanje nota, počeci nota, dinamika, intenzitet i boja zvuka nisu eksplicitno prikazani, a to su najbitnije značajke zvuka koje su potrebne kako bi se provela automatska transkripcija glazbe. U nastavku su detaljnije objašnjene navedene značajke.

2.2.1. Frekvencija i visina tona

Audio signal odnosno zvuk može biti vizualno prikazan kao valni oblik. Najjednostavniji valni oblik je sinusoida. Ukoliko se ciklus vala odnosno točke najveće i najmanje vrijednosti ponavljaju bez promjene onda kažemo da je val periodičan. Period vala je definiran kao vrijeme potrebno da val napravi jedan ciklus [5, str. 21] odnosno u slučaju sinusoide, val mora prijeći iz nule u najvišu točku, zatim proći nulu i otići u najnižu točku i vratiti se u nulu. Najveće odstupanje od prosječne vrijednosti vala predstavlja amplitudu signala. Još jedna karakteristika vala jest faza, a određuje odstupanje ciklusa vala od nultog vremena. Frekvencija predstavlja recipročni broj od perioda vala te se mjeri u Hertzima (Hz). Povećanjem frekvencije zvuk se čuje višlje i obrnuto. Ljudsko uho je u mogućnosti čuti frekvencije u rasponu od 20 Hz do 20 kHz. Frekvencija i visina tona su usko povezani odnosno to je subjektivan atribut zvuka kojem se u slučaju kompleksnijih zvučnih sastava teško je odrediti jednostavnu povezanost s frekvencijom, dok u slučaju čistog tona frekvencija i visina tona predstavljaju istu stvar npr. frekvencija 440 Hz predstavlja visinu tona A4, što se često naziva koncertna visina tona te ona predstavlja referencu visine tona prema kojoj se uštimavaju glazbeni instrumenti [5, str. 21-22]. Dvije frekvencije se smatraju sličnima ako se razlikuju s faktorom 2, npr. visina tona A3 (220 Hz), A4 (440 Hz) i A5 (880 Hz) se smatraju sličnim frekvencijama dok za visine tonova kažemo da se razlikuju za jednu oktavu. Oktava je u glazbi interval između dva tona kojima se frekvencije razlikuju za faktor 2, npr. za visine tonova

noti A3 (220 Hz), A4 (440 Hz) i A5 (880 Hz) kažemo da slični i da su razmaknuti za oktavu. Jedna oktava je podijeljena na 12 dijelova odnosno noti [6].

2.2.2. Počeci i trajanje nota

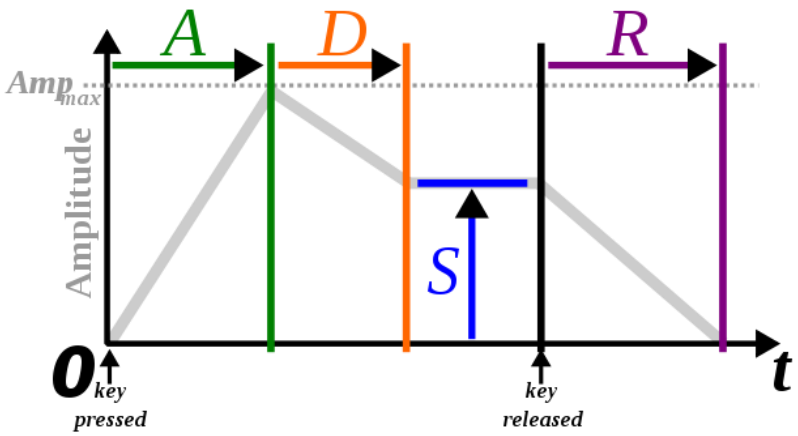
Kako bi se točno proveo postupak automatske transkripcije glazbe vrlo bitni faktori su počeci i trajanje pojedinih nota. Počeci nota (engl. *onset*) daju nam vremenske točke u kojima je određeni instrument započeo s reprodukcijom određene visine tona. Ukoliko se kvalitetno ne odrede ovakve značajke može doći do preklapanja nota i/ili praznim intervalima odnosno pauzama između nota u krajnjem izlazu određenog sustava. Trajanje nota također je bitno te je povezano sa završetcima nota (engl. *offset*). Kod slučaja trajanja note odnosno *offset* značajkama nije nužno potpuno precizno odrediti koliko je vrijeme odnosno točka završetka jer ni ljudsko uho nije u potpunosti u mogućnosti odrediti kraj trajanja note. Zato se trajanje nota obično otprilike određuje do početka iduće note ukoliko glasnoća same note previše opadne. Naravno to nije univerzalno pravilo jer u puno skladbi je namjerno odsvirano više noti odjednom kako bi se upotpunio zvuk te sami primjer toga su akordi koji se sastoje od barem tri tona izvedenih u isto vrijeme.

2.2.3. Dinamika i intenzitet

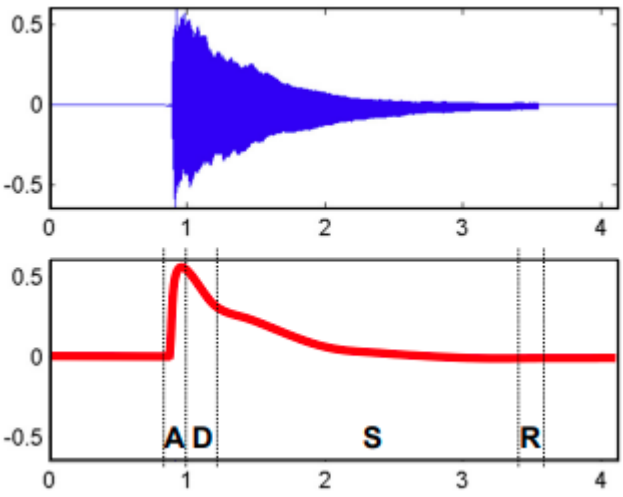
Još jedno od bitnih svojstava glazbe je dinamika. Dinamika predstavlja pojam koji nam govori kako se određene note trebaju svirati. Ukoliko se note sviraju jače obično to znači da ćemo te note glasnije čuti te kažemo da su one naglašene. Isto vrijedi i obrnuto odnosno za note koje se sviraju nježnije. U audio reprezentaciji dinamika se korelira s glasnoćom te, kao u navedenom odnosu frekvencije i visine tona u poglavlju 2.2.1, glasnoća je subjektivna mjera koja se korelira s mjerama intenziteta zvuka i snage zvuka, no glasnoća također ovisi i o drugim zvučnim karakteristikama kao što su trajanje i frekvencija [5, str. 24-26]. Generalno, snaga je mjera koja govori koliko se energije prenijelo, iskoristilo ili transformiralo i izražava se u Wattima (W) odnosno džulima po sekundi. Snaga zvuka nam govori koliko je energije u jedinici vremena odaslano iz zvučnog izvora u zrak prema svim smjerovima. Intenzitet zvuka nam govori koliko je snage zvuka odaslano po jedinici površine te se izražava u Wattima po kvadratnom metru ($\mathrm{W/m^2}$) odnosno u decibelima (dB). Postoje dva praga koji su bitni za intenzitet zvuk, a to su prag sluha (engl. *threshold of hearing, TOH*) i prag boli (engl. *threshold of pain, TOP*). *TOH* predstavlja najmanji intenzitet potreban kako bi ljudsko uho raspoznalo zvuk, a obično je intenziteta $\mathrm{I_{TOH}} \approx 10^{-12}\ \mathrm{W}\ /\ \mathrm{m^2}$, dok *TOP* predstavlja gornju granicu intenziteta koje ljudsko uho može raspoznati, a intenziteta je $\mathrm{I_{TOP}} \approx 10\ \mathrm{W}\ /\ \mathrm{m^2}$.

2.2.4. Boja zvuka

Pored navedenih karakteristika zvuka, još je jedan važan aspekt koji se naziva boja zvuka odnosno tona (engl. *timbre*). Boja zvuka predstavlja kvalitetu zvuka pomoću koje se razlikuju različiti instrumenti i glasovi čak i ako imaju jednake visine tonova i glasnoću [9]. Jedna od karakteristika boje zvuka je vremenska evolucija koja predstavlja određene faze razvijanja, održavanja i nestajanja tona. Navedene faze modeliraju se *ADSR* modelom koji predstavlja fazu početka tona, fazu smirivanja tona, fazu održavanja tona i fazu otpuštanja tona (engl. a*ttack, decay, sustain, release, ADSR*). *ADSR* model prikazan je slikom 2.1.. Tijekom *attack* faze zvuk se gradi i doseže svoju maksimalnu amplitudu te je ta faza često popraćena dodatnim šumom. U *decay* fazi zvuk se stabilizira i dolazi do stabilnog periodičkog uzorka. U *sustain* fazi energija zvuka je poprilično konstantna. *Release* faza predstavlja otpuštanje zvuka odnosno ton polako blijedi i približava se razini nečujnosti. Primjer *ADSR* modela na stvarnom primjeru C4 note odsvirane na klaviru prikazan je na slici 2.2..



Sl. 2.1. *ADSR* model



Sl. 2.2. Primjer *ADSR* modela

2.3. Formati zapisa glazbe u računalnom obliku

Simboličke reprezentacije odnosno formati zapisa glazbe opisuju glazbu po njezinim značajkama odnosno atributima, a u slučaju digitalnog formata još mogu biti računalno obrađivani. Bilo koji digitalni format podataka se može smatrati simboličkim jer sadrži konačan skup oznaka (abeceda) kojim se opisuju podaci. Tako u simboličke reprezentacije spada sva digitalizirana glazba, no takav format nije pogodan čovjeku za samostalno reproduciranje glazbe. Notni zapis i tablature predstavljaju „pravi“ simbolički zapis pomoću kojeg ljudi mogu rekreirati originalnu izvedbu određene pjesme. Navedena dva zapisa nisu od prevelike koristi samom računalu te su u tu svrhu stvoreni formati kao što su klavirni svitak (engl. *piano-roll*), MIDI format i MusicXML format.

2.3.1. Klavirni svitak (*piano-roll*)

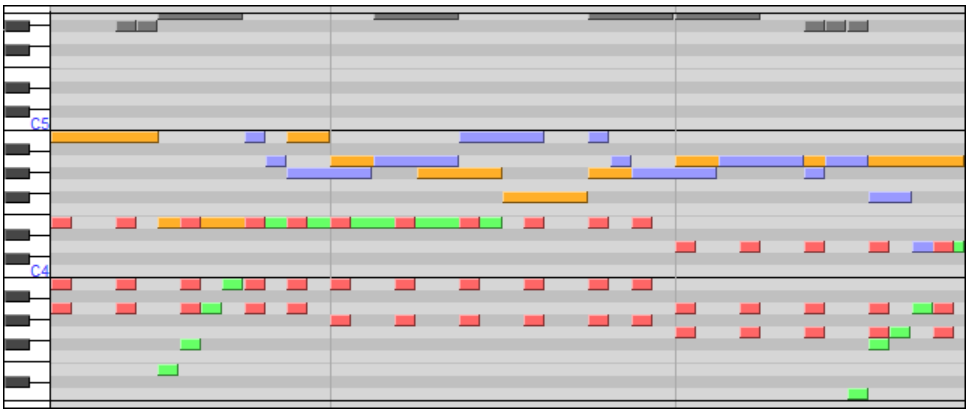
Krajem 19. stoljeća i početkom 20. klaviri koji su sami svirali su bili popularni. Ulaz takvih glazbala bio je svitak odnosno komad papira, prikazan na slici 2.3., koji je na sebi imao probušene rupe koje su predstavljale note i njihovo trajanje [5, str. 11]. Danas se pod tim pojmom podrazumijeva bilo koji prikaz informacija o notama koji sliči klavirnom svitku, vidi sliku 2.4.. U 2D prikazu klavirnog svitka horizontalna os predstavlja vrijeme dok se na vertikalnoj osi prikazuju visine tonova.

Sl. 2.3. Klavirni svitak

2.3.2. MIDI format

MIDI standard (engl. *Musical Instrument Digital Interface*) nastao je u razdoblju između 1981. i 1983. godine zbog velikog rasta korištenja elektroničkih glazbenih instrumenata. MIDI format za svaku notu sprema njezine značajke kao što su početak trajanja note, završetak trajanja note i intenzitet (u MIDI terminologiji još zvano i *velocity*). Bitno je naglasiti kako MIDI ne predstavlja direktno zvuk glazbe nego predstavlja način izvođenja glazbe odnosno način sviranja instrumenta. [5, str. 13] Oznaka note u MIDI formatu ima cijeli broj od 8 bita (vrijednosti od 0 do 127) koji

predstavlja visinu tona. Glavna značajka MIDI formata je ta da središnja nota odnosno nota C4 ima broj 60 te su oktave razmaknute za 12 brojeva npr. C5 je broj 72, C6 je broj 84. Brzina okidanja note (engl. *key velocity*) također je predstavljena 8-bitnim brojem i predstavlja intenzitet note. MIDI format podržava 16 kanala odnosno instrumenata koji su spremljeni kao 4-bitni broj. Format također sprema podatke kao što su tempo i takt (engl. *beats per minute, BPM*). Vizualni prikaz MIDI formata sličan je piano-rollu, vidi sliku 2.4..



Sl. 2.4. Vizualizacija MIDI formata

2.3.3. MusicXML format

MusicXML (engl. *Music Extensible Markup Language*) je označni jezik baziran na XML formatu (engl. *eXtensible Markup Language*) te predstavlja univerzalni format za spremanje glazbe. Strukturiran je hijerarhno kao i svi ostali označni jezici te sadrži skup svojih oznaka kao što su: *note, pitch, step, octave, duration, measure*, itd. [8]. Sprema podatke o svakoj noti kao što su visina tona i trajanje te strukturne glazbene podatke o cijeloj pjesmi. Primjer jedne note zapisane u MusicXML-u prikazan je na slici 2.5..

Linija ***Kôd***

```
1:      <note>
2:        <pitch>
3:          <step>C</step>
4:          <octave>4</octave>
5:        </pitch>
6:        <duration>4</duration>
7:        <type>whole</type>
8:      </note>
```

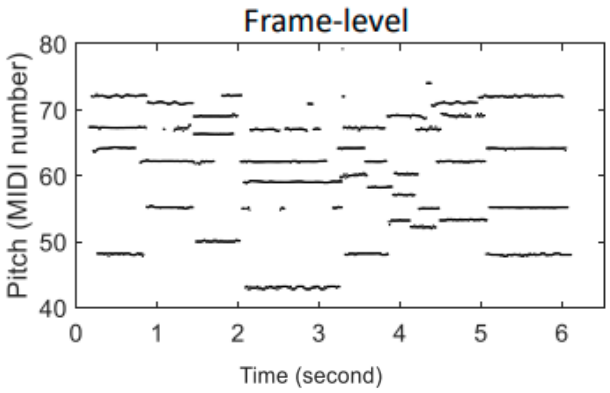
Sl. 2.5. Kôd primjer note u MusicXML-u

2.4. Razine automatske transkripcije glazbe

Kako bi se pravilno i kvalitetno provela automatska transkripcija glazbe potrebno je problem rastaviti na manje korake odnosno faze. Navedeno u poglavlju 2.1, još je 1977. godine dr.sc. Martin Piszczalski u svom radu podijelio proces *AMT*-a u tri faze. Ideja je dan danas ista no razvojem tehnologije i alata došlo je do malih promjena u navedenim fazama procesa. Godine 2018. grupa profesora i znanstvenika, predvođena Emmanouilom Benetosom, izdala je rad pod imenom *Automatic Music Transcription: An Overview* [9] u kojem su postavljene smjernice odnosno nove verzije navedenih faza *AMT*-a. Faze *AMT*-a prema novim standardima tehnologije su okvirno sljedeće: razina okvira (engl. *Frame-level*), razina nota (engl. *Note-level*), razina toka (engl. *Stream-level*) i razina notacije (engl. *Notation-level*). Iako su faze procesa *AMT*-a promijenjene, svrha je i dalje ostala ista, a to je pretvorba zvučnog glazbenog zapisa u neku od vrsta glazbenih notacija odnosno formata. Navedene faze detaljnije su objašnjene u sljedećim potpoglavljima.

2.4.1. *Frame-level* transkripcija

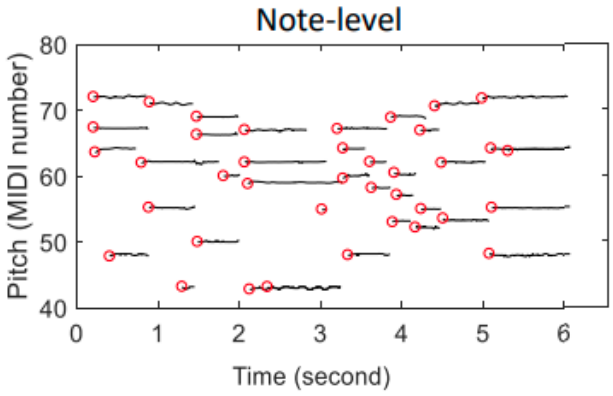
Frame-level razina transkripcije još se zove i procjena više visina tonova (engl. *Multi-Pitch Estimation, MPE*) te predstavlja procjenu broja nota i visina tonova pojedinih nota koje su istovremeno prisutne u svakom vremenskom okviru [9, str. 3]. Zvučni zapis se dijeli na jednake vremenske intervale odnosno okvire u rasponu od oko 10 ms. Metode na ovoj razini transkripcije ne formiraju koncepte glazbenih nota i jako rijetko modeliraju više glazbene strukture. Neke od metoda u ovoj fazi su: tradicionalno procesiranje i analiza signala, modeliranje pomoću vjerojatnosti, metode bazirane na Bayesovim mrežama vjerojatnosti, množenje nenegativnih matrica i korištenje neuronskih mreža odnosno strojnog učenja. Sve navedene metode imaju svoje prednosti i nedostatke te sveukupno istraživanje područja AMT-a nije konvergiralo u smjeru jednog pristupa rješavanja problema. Na primjer, tradicionalno procesiranje signala (baziranog na FFT, konstantnoj Q transformaciji, ...) je jednostavno i brzo i lakše generalizira različite instrumente, dok neuronske mreže obično dobivaju bolje rezultate na određenim instrumentima na kojima su trenirane. Bayesovi pristupi omogućuju sveobuhvatno modeliranje procesa generiranja zvuka, no takvi modeli mogu biti komplicirani i spori. Primjer izlaza podataka nakon *frame-level* razine prikazan je na slici 2.6., gdje svaka točkica predstavlja predviđenu visinu tona i njegovo trajanje u sekundama unutar jednog okvira.



Sl. 2.6. Primjer izlaza iz *frame-level* razine transkripcije

2.4.2. *Note-level* transkripcija

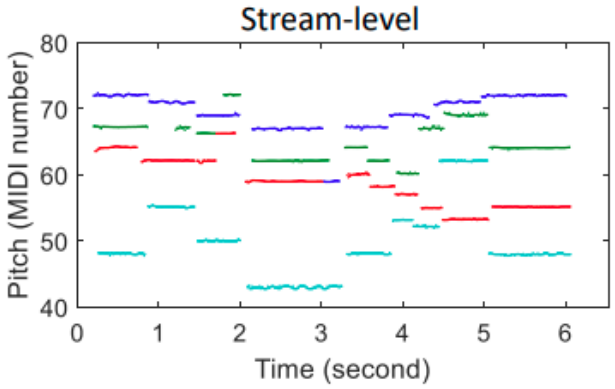
U sklopu *note-level* transkripcije provode se predikcije i analiza dobivenih izlaza iz *frame-level* faze. U ovoj fazi svaka predviđena visina tona se spaja u vremenskoj domeni u note. Note odnosno njihove MIDI reprezentacije su najčešće karakterizirane s prethodno navedenim svojstvima, a to su: visina tona, početak note i kraj note. Kako krajevi nota znaju biti nesigurni, dosta često se u ovoj fazi zanemaruju odnosno njihovo određivanje ne mora nužno biti sasvim točno. Metode koje su korištene u ovoj fazi su često filtriranje pomoću medijan vrijednosti, korištenje skrivenih Markovljevih modela (engl. *Hidden Markov Models, HMM*) i uporaba neuronskih mreža [9, str. 4]. Ovi načini procesiranja često predviđaju pogrešno zalutale note koje dijele harmonike s točno predviđenim notama. Neke od predloženih metoda u ovoj fazi koje nisu učestale predlažu korištenje i analizu međusobnih interakcija nota kroz vjerojatnosni model spektra ili čak i jezične glazbene modele (engl. *music language models*). Neki od načina su još i predviđanje nota direktno iz audio zapisa, a ne naknadna obrada izlaza iz *frame-level* faze. Na slici 2.7. vidi se primjer izlaza iz *note-level* razine transkripcije, gdje se počeci nota predstavljaju crvenom kružnicom koju prati crna linija koja predstavlja visinu tona i predviđeno trajanje note.



Sl. 2.7. Primjer izlaza iz *note-level* razine transkripcije

2.4.3. *Stream-level* transkripcija

Stream-level faza transkripcije također je zvana i faza toka više visina tonova (engl. *Multi-Pitch Streaming, MPS*). U ovoj fazi se ciljaju grupe predviđenih visina tonova odnosno nota te se grupiraju u tokove, gdje svaki tok predstavlja neki od instrumenata i/ili glasova što je podosta povezano s problemom razdvajanja instrumenata (engl. *audio source separation*) [9, str. 4]. U usporedbi s *note-level* razinom, konture visina tonova unutar svakog toka su puno duže od trajanja jedne note i sadrže mnoge nepravilnosti koje su uzrokovane tišinom, šumom i isprekidanim promjenama frekvencija. Zbog toga unutar *note-level* faze nije moguće grupirati visine tonova u dugačke i neisprekidane konture kao što je to moguće u *stream-level* fazi. Jedno od glavnih svojstava glazbe i zvuka koje je potrebno kako bi se uspješno provela ova faza transkripcije je boja tona. Note u tokovima najčešće reprezentiraju instrumente odnosno može se reći da je tok zapravo zvuk jednog instrumenta, a već je prethodno objašnjeno u poglavlju 2.2.4 kako se prema boji tona mogu odrediti različiti instrumenti. U literaturi se ova faza također često naziva faza praćenja boje zvuka (engl. *timbre tracking*) odnosno faza izdvajanja instrumenata (engl. *instrument tracking*). Problem unutar ove faze je što se vrijeme još uvijek mjeri u sekundama umjesto u taktu, visine tonova još su uvijek predstavljene kao MIDI broj umjesto prigodnih formata odnosno imena nota te koncepti takta, ključa i harmonije nisu prisutni. Na slici 2.8. prikazan je primjer izlaza is *stream-level* razine transkripcije gdje svaka boja predstavlja predviđene note nekog instrumenta.

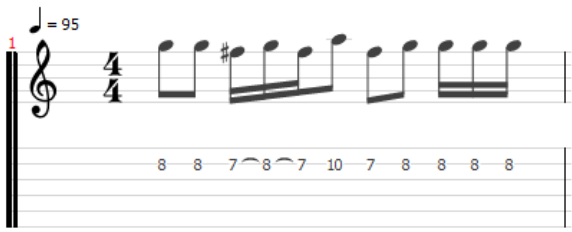


Sl. 2.8. Primjer izlaza iz *stream-level* razine transkripcije

2.4.4. *Notation-level* transkripcija

Cilj *notation-level* razine transkripcije je pretvorba glazbenog zapisa u čovjeku čitljiv zapis odnosno notni zapis. Transkripcija na ovoj razini zahtijeva dublje poznavanje i analizu glazbenih struktura kao što su harmonici, ritam i strukture tokova [9, str. 4]. Strukture harmonika kao što su ključevi i akordi utječu na note procijenjene iz MIDI visina tonova, strukture ritma kao što je takt

mogu doprinijeti određivanju trajanja pojedinih nota te strukture tonova utječu na već spomenuti problem određivanja pripadajućih nota određenog instrumenta. Neke od predloženih metoda za ovu razinu transkripcije su: kvantizacija vremena, razdvajanje glasova i *end-to-end* neuronske mreže. Problem postojećih softverskih rješenja je taj što krajnji rezultati većinom nisu zadovoljavajući i nije jasna razlika u glazbenim strukturama koje su predviđene prilikom cijelog procesa transkripcije. Primjer izlaza *notation-level* razine bio bi notni ili tabularni zapis prikazan na slici 2.9..



Sl. 2.9. Primjer izlaza iz *motation-level* razine transkripcije

3. Analiza signala

Prvi ključan korak u procesu automatske transkripcije glazbe je dobro poznavanje ulaza u sustav odnosno dobar matematički opis i analiza melodije koju želimo transkriptirati. U ovom poglavlju bit će detaljnije opisane neke od metoda navedenih za *frame-level* razinu transkripcije. Bit će objašnjene metode za detekciju frekvencije te određivanje visine tona, kao i vizualizacija izlaza takvih metoda. Prikazati će se kako se u sve to uklapa polifonijska glazba i harmonija nota te jednostavnim programskim primjerima provesti navedene metode za analizu audio signala.

3.1. Spektrogram

Spektrogram je vizualna reprezentacija spektra frekvencija ulaznog signala raspoređenih u vremenskoj domeni [10]. Spektrogram je u stvari dvodimenzionalni graf koji na horizontalnoj osi prikazuje vrijeme, a na vertikalnoj frekvencije. Frekvencije se prikazuju bojama ili nijansama određene boje u ovisnosti o njihovoj gustoći. Tako na primjer područja spektra gdje se nalazi velika gustoća frekvencija su obično označene izraženijim svijetlim bojama dok se područja male gustoće frekvencija prikazuju tamnijim bojama. Spektrogram se najčešće dobije tako što se na ulaznom signalu provede neki od oblika Fourierovih transformacija, opisanih u poglavlju 3.2, na prethodno određenim jednakim intervalima odnosno okvirima na koje je podijeljen ulazni signal. Spektrogram se u programskom jeziku *Python* može dobiti iz biblioteke *librosa* [11] naredbom librosa.display.specshow te se primjer izlaza navedene naredbe vidi na slici 3.4..

3.2. Fourierova transformacija

Fourierova transformacija je jedna od najbitnijih operacija u primijenjenoj matematici i analizi signala. Cilj Fourierove transformacije je pretvorba ulaznog signala iz vremenske domene u frekvencijsku domenu [12]. Frekvencijska domena predstavlja signal kao superpoziciju sinusoida s različitim magnitudama, frekvencijama i fazama. Izlaz Fourierove transformacije najčešće se prikazuje kao histogram odnosno prikazuje se distribucija frekvencija u određenom frekvencijskom rasponu. Fourierova transformacija definira se kao:

$$X(i\omega) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t}dt \tag{3-1}$$

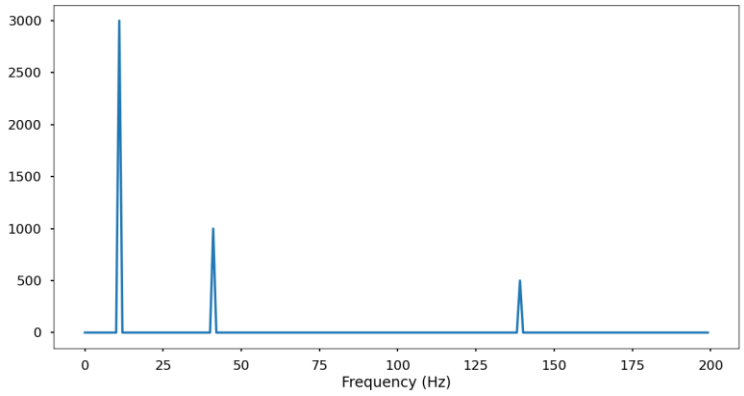
gdje je: *x(t)* – ulazni signal u vremenskoj domeni, $e^{-i\omega t}$ – kompleksna harmonijska funkcija, a *dt* – derivacija to varijabli *t*. Fourierova transformacija može se dobiti pomoću *Python* biblioteke *scipy* [13] naredbom *scipy.fft*. Primjer kôda izvršavanja Fourierova transformacije nad tri

sinusoidna signala (x) frekvencija: 11 Hz, 41 Hz i 139 Hz, prikazan je na slici 3.1., a izlaz takvog programa na slici 3.2..

Linija ***Kôd***

```
1:      X = fft(x)
2:      magnitudes = np.abs(X)
3:      f = np.linspace(0, sr, len(magnitudes))
4:      plt.figure(figsize=(13, 5))
5:      plt.plot(f[:200], magnitudes[:200])
6:      plt.xlabel('Frequency (Hz)')
7:      plt.show()
```

Sl. 3.1. Kôd primjer Fourierove transformacije



Sl. 3.2. Izlaz koda za izračun Fourierove transformacije

3.2.1. *Short-time* Fourierova transformacija

Ukoliko se ulazni signal podijeli na dovoljno male intervale, na kojima se zatim provede Fourierova transformacija, dobijemo distribuciju frekvencija u svakom pojedinom intervalu te se takav izlaz može prikazati kao prethodno opisani spektrogram. Takva Fourierova transformacija se naziva *Short-time Fourier Transform* (*STFT*) [14]. U svrhu *AMT* procesa provodi se ovakav tip Fourierove transformacije, jer ne bi imalo smisla provoditi normalnu Fourierovu transformaciju na cijelim skladbama odnosno pjesmama. *STFT* se definira kao:

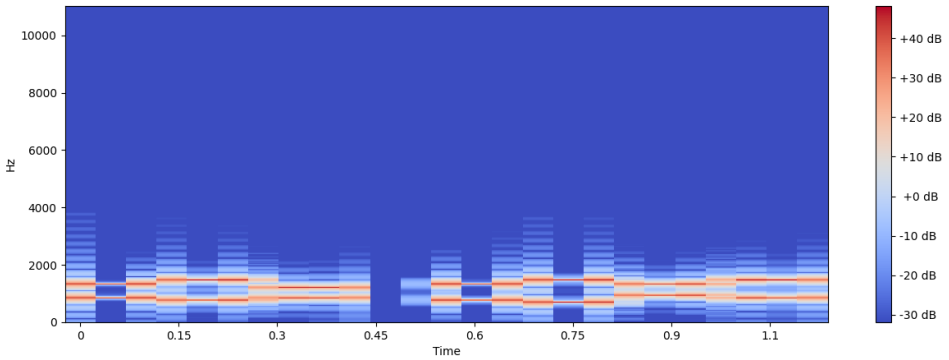
$$X(\tau, i\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-i\omega t}dt \tag{3-2}$$

gdje je: *w(τ)* – funkcija prozora odnosno intervala (engl. *window function*), *x(t)* – ulazni signal u vremenskoj domeni, $e^{-i\omega t}$ – kompleksna harmonijska funkcija, a *dt* – derivacija to varijabli *t*. Primjer kôda za izračun *Short-time* Fourierove transformacije nad ulazom koji predstavlja zvukove otipkavanja brojeva na telefonu prikazan je na slici 3.3., a njegov izlaz u obliku spektrograma prikazan je na slici 3.4..

Linija ***Kôd***

```
1:      X = librosa.stft(x, n_fft=2048, hop_length=1024)
2:      spectogram = librosa.amplitude_to_db(abs(X))
3:      plt.figure(figsize=(15, 5))
        librosa.display.specshow(spectogram, sr=sr, hop_length=1024,
4:      x_axis='time', y_axis='linear')
5:      plt.colorbar(format='%+3.0f dB')
6:      plt.show()
```

Sl. 3.3. Kôd primjer *Short-time* Fourierove transformacije



Sl. 3.4. Izlaz koda za izračun *Short-time* Fourierove transformacije

3.3. Konstantna Q transformacija

Konstantna Q transformacija (engl. *constant-Q transform*, *CQT*) transformira, kao i Fourierova transformacija, ulazni signal u frekvencijsku domenu [15]. *CQT* se može promatrati kao niz filtera f_k, koji su logaritamski razmaknuti u frekvenciji, gdje δf_k predstavlja spektralnu širinu k-tog filtera, koje je jednaka umnošku prijašnje širine filtera, odnosno:

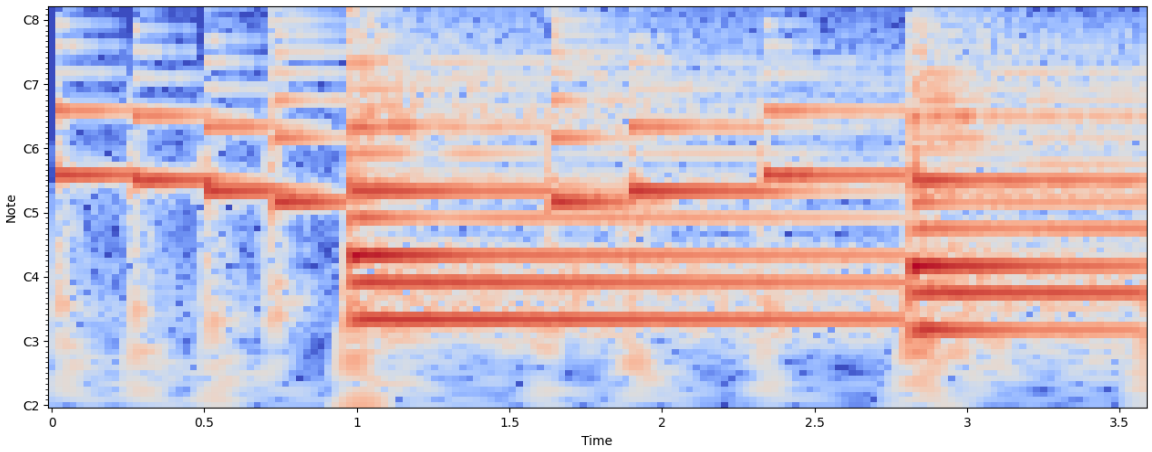
$$\delta f_k = 2^{1/n} * \delta f_{k-1} = \left(2^{1/n}\right)^k * \delta f_{min} \tag{3-2}$$

gdje je: δf_k – propusnost odnosno širina k-tog filtera, f_{min} – centralna frekvencija najmanjeg filtera, n – broj filtera unutar jedne oktave. Bitna mjera vezana za *CQT* je faktor kvalitete koji određuje omjer između centralne frekvencije i frekvencije specifične spektralne komponente, a označava se i izračunava kao $Q = \frac{f_k}{\delta f_k}$. Bitno je i napomenuti kako je *CQT* pogodnija od Fourierove transformacije u pogledu obrade glazbe. Prednost *CQT* je ta što može niže note razlučiti jednako dobro kao i visoke zato što centralne frekvencije mogu biti izabrane kako bi se bolje podudarale s polutonovima kakvi postoje i u kromatskoj ljestvici [16]. Također još jedna prednost logaritamsko raspodijeljene frekvencijske osi je ta što harmonici svih noti imaju jednaku povezanost s temeljnom notom. Primjer kôda konstantne Q transformacije prikazan je na slici 3.5. te je, kao i u prethodnim primjerima, korištena biblioteka *librosa*. Kao ulaz odabrana je melodija odsvirana na klaviru, a izlaz je prikazan na slici 3.6..

Linija ***Kôd***

```
1:      fmin = librosa.midi_to_hz(36)
2:      CQT = librosa.cqt(x, sr=sr, fmin=fmin, n_bins=75, hop_length=512)
3:      logCQT = librosa.amplitude_to_db(numpy.abs(CQT))
4:      plt.figure(figsize=(15, 5))
        librosa.display.specshow(logCQT, sr=sr, x_axis='time',
5:      y_axis='cqt_note', fmin=fmin, cmap='coolwarm')
6:      plt.show()
```

Sl. 3.5. Kôd primjer konstantne Q transformacije



Sl. 3.6. Izlaz koda za konstantnu Q transformaciju

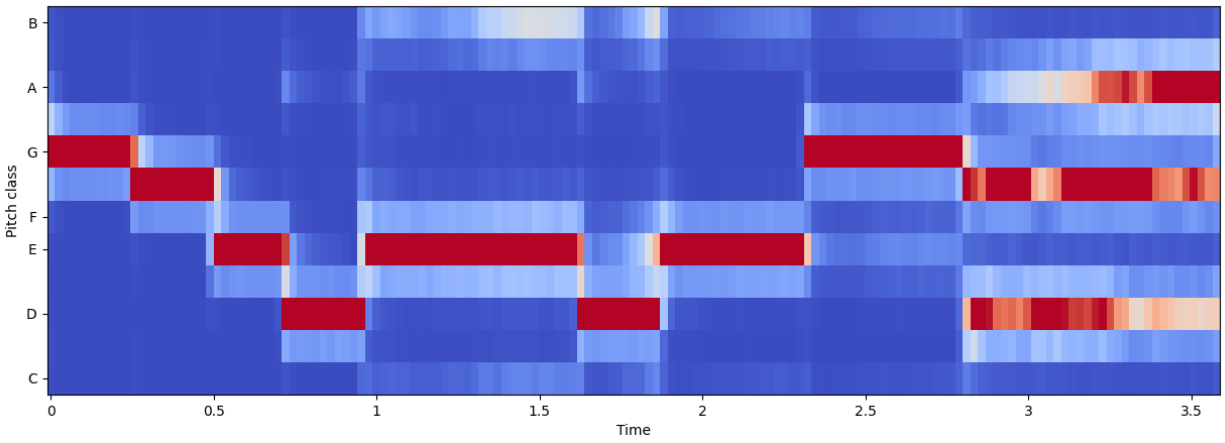
3.3.1. Chromagram

Chroma vektor ili *chromagram* je vektor od 12 elemenata koji govore koliko energije ima u svakoj od klasa visina tonova (C, C#, D, D#, ..., B) [17]. Elementi su podijeljeni u skladu s oktavama. Glavno svojstvo *chromagrama* je ta da je u stanju uhvatiti harmonijske i melodijske karakteristike glazbe dok je u isto vrijeme vrlo robustan odnosno ne promjenjiv na promjene u boji tona i odabiru instrumenata. Primjer kôda za prikaz *chromagrama* nalazi se na slici 3.7., za ulaz je odabrana ista melodija kao i u poglavlju 3.3, a izlaz je moguće vidjeti na slici 3.8.. Na slici se također znatno jasnije vide note odsvirane u ulaznoj melodiji nego što se vide na slici 3.6. te je baš zbog toga *chromagram* puno prilagodniji tip spektrograma od standardnog u slučaju obrade glazbe.

Linija ***Kôd***

```
1:      chromagram = librosa.feature.chroma_stft(y=x, sr=sr, hop_length=512)
2:      plt.figure(figsize=(15, 5))
        librosa.display.specshow(chromagram, x_axis='time', y_axis='chroma',
3:      hop_length=512, cmap='coolwarm')
4:      plt.show()
```

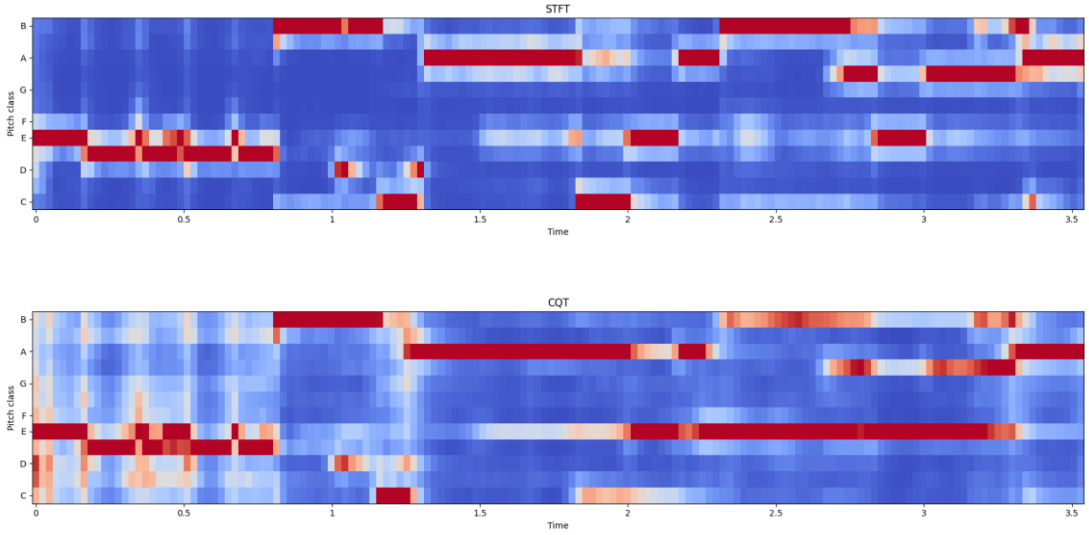
Sl. 3.7. Kôd primjer za prikaz *chromagrama*



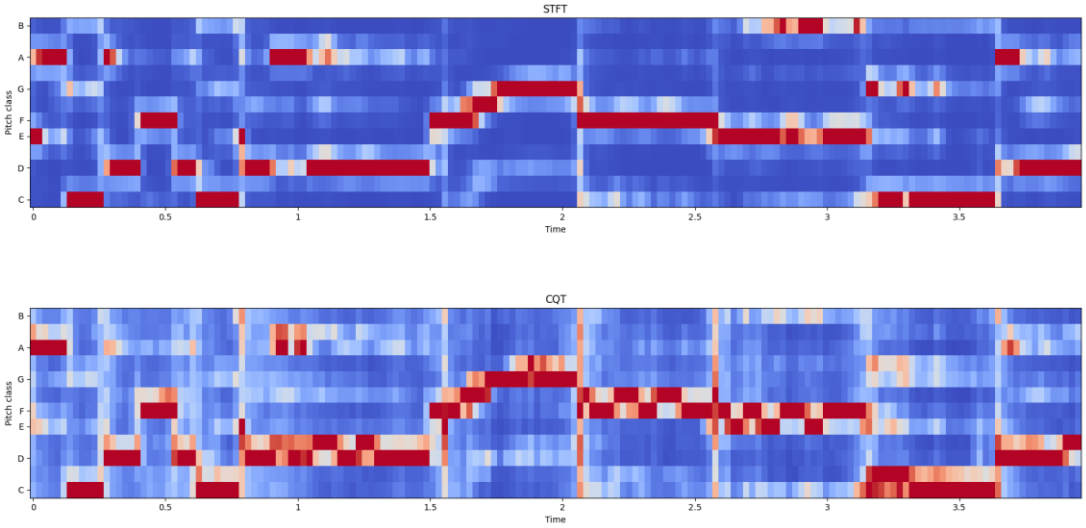
Sl. 3.8. Prikaz *chromagrama*

3.4. Visina tona

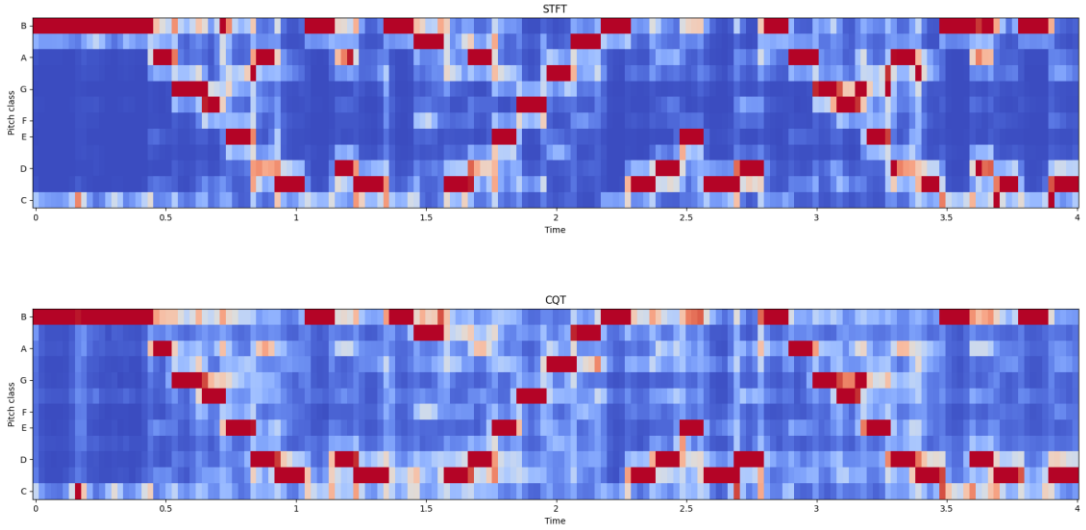
Jedan od glavnih i najosnovnijih dijelova automatske transkripcije glazbe je određivanje visine tona odnosno određivanje nota. Ovaj dio procesa je povezan s prethodno opisanim transformacijama (navedenim u poglavljima 3.2.1 i 3.3) te se takvi postupci određivanja tona nameću od početka nastanka problema *AMT*-a. Obje transformacije u ovom koraku *AMT*-a daju ono što je i potrebno, a to su predikcije frekvencija koje se koreliraju s visinama tonova. U ovom poglavlju usporedit ćemo *Short-time* Fourierovu i konstantnu Q transformaciju. Uspoređivat će se izlazi u obliku *chromagrama* te će se u oba slučaja uzimati isti parametri kao što su duljina intervala na kojima se određuje transformacija i brzina uzorkovanja. Na izlazima gornji *chromagram* odnosit će se na *STFT* dok će se donji odnositi na *CQT*. U svrhu usporedbe uzete su melodije odsvirane na klaviru, električnoj gitari i violini. Zbog jednostavnosti i boljeg prikaza u melodijama su prisutni samo zvukovi navedenih instrumenata, bez dodatnih instrumenata i glasova. Na slici 3.9. primjećuje se kako je *Short-time* Fourierova transformacija točnije otkrila visine tonova odnosno note unutar melodije klavira nego konstantna Q transformacija, kao i početke završetke istih. Na slici 3.10., koja predstavlja melodiju električne gitare, obje metode su podjednako otkrile početke i završetke noti no pomoću *CQT* nije uspješno otkrivena visina tona kao u *STFT*. Na slici 3.11. odnosno na *chromagram*-u melodije violine, obje transformacije su dosta slično otkrile visine tonova te se u svakoj vide različita mala odstupanja, a što se tiče početaka i završetaka nota, obje transformacije su podjednako pretpostavile navedeno. U svakom od ova tri primjera moguće je primijetiti kako je *CQT* više osjetljiva metoda na šum i smetnje u melodiji. To se primjećuje u bijelim i narančastim područjima kojih u primjerima sa *STFT* ima značajno manje. Obje transformacije imaju svoje prednosti i mane te korištenje određene ovisi o kontekstu i žanru melodije odnosno skladbe.



Sl. 3.9. Usporedba *STFT* i *CQT* na melodiji klavira



Sl. 3.10. Usporedba *STFT* i *CQT* na melodiji električne gitare



Sl. 3.11. Usporedba *STFT* i *CQT* na melodiji violine

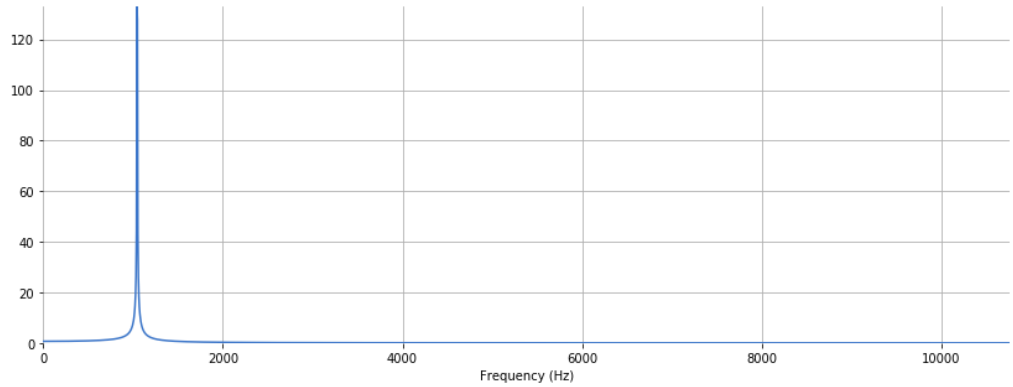
3.5. Polifonija

Unutar područja automatske transkripcije glazbe pojmovi monofonija i polifonija se često uspoređuju kako bi se opisale dvije vrste signala. Pod pojmom monofonija smatra se sav oblik glazbe koji sadrži samo jedan instrument ili glas odnosno u audio snimci prisutan je samo jedan izvor zvuka [16, str. 41-42]. Polifonija predstavlja oblik glazbe koji sadrži više od jednog glasa i/ili instrumentala odnosno u audio snimci je prisutno više izvora zvuka. Predviđanje noti monofonijske glazbe trivijalan je postupak koji se bazira na osnovnoj obradi i analizi signala. Takvi postupci i metode objašnjeni su i prikazani u poglavlju 3.4. Predviđanje polifonijske glazbe (velike većine glazbe u svijetu) zahtijevan je i težak proces koji još uvijek nije kompletno usavršen. U novije vrijeme velika većina metoda za transkripciju polifonijske glazbe oslanja se na strojno učenje i umjetnu inteligenciju no postoje i neke druge metode koje su se pokazale kao dobar izbor kao što je na primjer množenje nenegativnih matrica. Detaljnije o navedenim metodama objašnjeno je u poglavljima 5.1 i 5.2. Problemi polifonijske glazbe za kojih još ne postoji konkretno rješenje su mnogi no neki od njih su: problem detekcije većeg broja visina tonova odjednom, preklapanje istih ili sličnih visina tonova nastalih od različitih instrumenata odnosno izvora, preklapanje početaka noti nastalih od različitih izvora, problem razdvajanja izvora, problem određivanja međusobne korelacije predviđenih nota različitih izvora, problem pojave harmonika (detaljnije objašnjen u poglavlju 3.6) itd. Zbog polifonije moguće je da je problem *AMT*-a nerješiv to jest da pripada *AI-complete* tipovima problema, a pod time se smatraju problemi koji su računski preteški za doći do kompletnog i točnog rješenja, kao na primjer kreiranje generalne umjetne inteligencije [18].

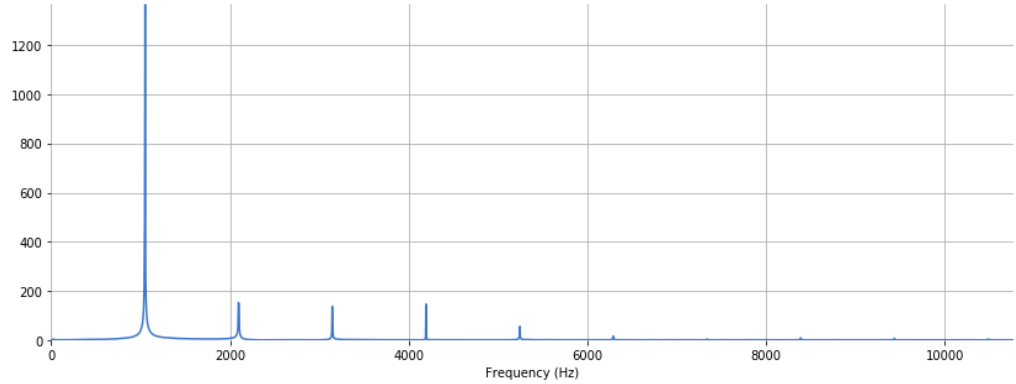
3.6. Harmonija

Jedan od osnovnih svojstava glazbe, a vrlo bitnih za proces *AMT*-a, je harmonija. Harmonija je pojava odnosno zvuk koji nastaje kada se dvije ili više visina tonova izvode u isto vrijeme [16, str. 42-43]. Pripada okomitom aspektu glazbe odnosno predstavlja ono što se događa u glazbi u jednom trenutku ili u vrlo kratkom intervalu, a nastaje zbog vodoravnog aspekta koji predstavlja ono što se događa u određenom vremenskom intervalu kao što je to na primjer ritam ili melodija. Kako bi se harmonija bolje definirala važno je razumjeti pojam interval. Interval u glazbenom smislu predstavlja razliku između dvije visine tonova [5, str. 243-246]. Najosnovniji interval u glazbi predstavlja oktava koja je definirana kao razlika između dvije visine tonova s udaljenosti od pola odnosno duplo nego temeljna frekvencija te se obično dijeli na 12 dijelova. Za dvije note npr. notu C2 i C3 kažemo da su razmaknute za jednu oktavu odnosno između njih postoji 12

skokova između noti. Najosnovniji primjer harmonije su harmonici odnosno pojava nastanka jedne ili više noti koje su za jednu ili više oktava udaljene od note koja se inicijalno želi izvesti. Primjer harmonika generiranog sviranjem jedne note na instrumentu prikazan je na slici 3.13., na kojoj se točno vidi pojava frekvencija udaljenih za duplo više odnosno za pola manje od inicijalne frekvencije dok se na slici 3.12. vidi prikaz računalno generiranog čistog tona koji se posjeduje karakteristike harmonika već samo sadrži temeljnu frekvenciju.



Sl. 3.12. Prikaz distribucije frekvencija računalno generiranog čistog tona



Sl. 3.13. Prikaz distribucije frekvencija tona odsviranog na fizičkom instrumentu

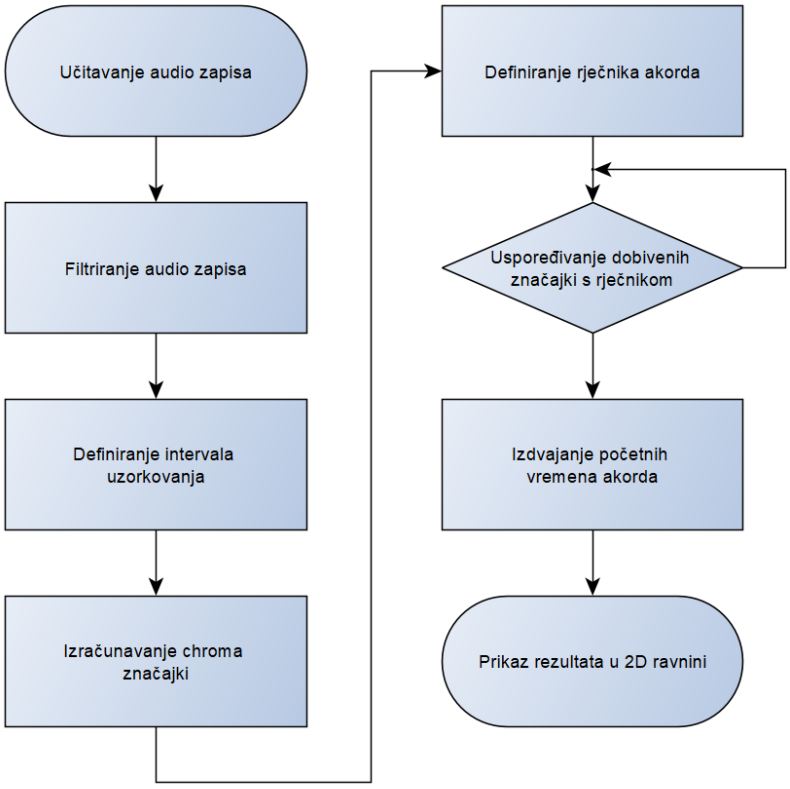
Za postupak *AMT*-a harmonija u nekim primjenama može biti korisna informacija, dok u nekim se može smatrati kao smetnja. Na primjer, u sklopu otkrivanja melodije koja sadrži određenu vrstu solaže (noti odsviranih jednu po jednu za redom) harmonija odnosno konkretno harmonici predstavljaju smetnju jer se očitavaju note duplo viših ili manjih frekvencija zbog čega dolazi do krivo predviđenih noti u istom vremenskom intervalu. Slučaj kada je harmonija poželjna je predviđanje akorda. Tada se pomoću raznih intervalnih struktura, kao što su to na primjer razmaci unutar kvintnog kruga, mogu odrediti glazbene strukture to jest glazbena povezanost noti unutar melodije baš zbog tih pravilnih intervala između dvije ili više visina tonova.

3.7. Detekcija akorda

Harmonija se u glazbi odnosi na izvođenje više različitih noti odjednom kako bi se stvorio izražajniji dojam kod slušatelja. Jedan od najosnovnijih oblika izvođenja više različitih nota odjednom su akordi. Akordi su gradivne glazbene jedinice koje se najčešće sastoje od tri ili više nota [5, str. 241-243]. Akordi mogu biti sastavljeni i od dvije različite note no to je često kontroverzna tema unutar teorije glazbe. Akordi su građeni na temelju intervala (objašnjenih u poglavlju 3.6) odnosno matematički opisanog odnosa između dvije note. Teorija intervala izlazi van okvira ovog rada te neće biti obrađena u sklopu istog. Dovoljno je znati da se glavna nota po kojoj akord dobiva ime zove korijenska nota te da su ostale note (u ovom slučaju ostale dvije) određene pomoću nekih od intervala (npr. terca, kvinta, ...). Akordi unutar problema *AMT*-a pripadaju podskupu harmonijskih problema, no može se reći da i sami stvaraju svoj problemski podskup. Jedan od značajnijih izazova pri detekciji akorda je taj što se tom problemu ne može pristupiti tradicionalno kao što se pristupalo u problemu određivanja visine tonova (poglavlje 3.4). Akordi se smatraju kao više nota izvedenih odjednom no to u stvarnosti nije tako. Prilikom izvođenja akorda na bilo kojem instrumentu gotovo je nemoguće izvesti sve potrebne note odjednom. To se događa zbog ljudske nepreciznosti i nepreciznosti samog instrumenta. Na primjer, akord odsviran na gitari, iako on ljudskom uhu zvuči kao tri note odsvirane odjednom, zapravo se sastoji od 3 note odsvirane jedna za drugom, koje se preklapaju, u vrlo kratkom periodu. Razlog tome je taj što je gotovo fizički nemoguće udariti tri žice odjednom. To ljudima ne predstavlja problem jer je razmak između noti dovoljno malen da bi bio neprimjetan, no kada se takav zvuk prenese u računalni audio zapis, onda nastaju problemi. Rješenje ovog problema rješava se ili dobrom kalibracijom intervala u kojima se uzorkuje *audio* zapis ili korištenjem kompleksnih metoda kao što su skriveni Markovljevi modeli ili strojno učenje. Kalibracija intervala uzorkovanja je jeftin i ne previše zahtjevan postupak no s njime se najčešće ne može generalizirati više instrumenata i žanrova, dok ostale navedene kompleksnije metode bolje generaliziraju u zamjenu za veću procesorsku potražnju. U većini metoda za detekciju akorda postoje dva koraka. U prvom koraku *audio* zapis se pretvara u niz audio značajki koje opisuju harmonijske veze unutar zvuka, najčešće u obliku *chroma* značajki opisanih u poglavlju 3.3.1. U drugom koraku provode se metode za detekciju odnosno uspoređivanje dobivenih značajki s već ustanovljenim značajkama koje opisuju strukturu akorda. Jedan od najjednostavnijih načina detekcije akorda je postupak detekcije akorda pomoću predložaka, koji je detaljnije objašnjen u sljedećem potpoglavlju (3.7.1).

3.7.1. Algoritam za detekciju akorda metodom predloška

Detekcija akorda metodom predloška predstavlja najjednostavniji oblik detekcije akorda. Glavna karakteristika ove metode je da se dobivene značajke uspoređuju s već poznatim značajkama o akordima te se traži najbliže podudaranje između navedenih. U svrhu ovog rada kreiran je algoritam u programskom jeziku *python* pomoću kojeg je provedena ova metoda detekcije akorda. Algoritam je podijeljen na osam dijelova koji su zasebno opisani, a dijagram koraka algoritma može se vidjeti na slici 3.14..



Sl. 3.14. Dijagram tijeka algoritma za detekciju akorda

U prvom dijelu algoritma se pomoću biblioteke *librosa* učitava audio zapis u varijablu *y* dok se frekvencija uzorkovanja toj *audio* zapisa sprema u varijablu *Fs*. Kôd ovog dijela vidi se na slici 3.15..

Linija ***Kôd***

```
1:      file = "test.wav"
2:      y, Fs = librosa.load(file, sr=None, mono=True, duration=None)
```

Sl. 3.15. Kôd primjer za učitavanje audio zapisa

U drugom koraku algoritma pristupa se filtriranju učitanog *audio* zapisa. Za filtriranje prvo je potrebno dizajnirati filter, a to se radi pomoću naredbe *butter* koja kao ulaz uzima vrijednost granične frekvencije i red filtera koji definira kompleksnost prijenosne funkcije filtera. Navedena naredba vraća koeficijente *a* i *b* koji definiraju prijenosnu funkciju filtera, koji se prosljeđuju zajedno s učitanim *audio* zapisom u funkciju *lfilter*, koja primjenjuje dizajnirani filter. Kôd ovog koraka algoritma vidi se na slici 3.16..

Linija ***Kôd***

```
1:      def lowpass_filter(data, cutoff_frequency, fs, order=5):
2:          half_fs = 0.5 * fs
3:          normal_cutoff = cutoff_frequency / half_fs
4:          b, a = butter(order, normal_cutoff, btype='low', analog=False)
5:          y = lfilter(b, a, data)
6:          return y
7:
8:      cutoff_frequency = 1250
9:      y = lowpass_filter(y, cutoff_frequency, Fs)
```

Sl. 3.16. Kôd primjer za filtriranje audio zapisa

U trećem koraku algoritma definira se veličina intervala uzorkovanja odnosno definira se veličina pojedinog intervala kao varijabla *window_size* i definira se veličina skoka (razmak između dva uzorka) kao varijabla *hop_length*. Kôd ovog dijela algoritma vidi se na slici 3.17.. Varijable koje su definirane su tih vrijednosti jer se tijekom testiranja algoritma ustanovilo kako te vrijednosti daju najbolje rezultate.

Linija ***Kôd***

```
1:      window_size = 4096
2:      hop_length = 2048
```

Sl. 3.17. Kôd primjer za definiranje veličine intervala uzorkovanja

U četvrtom koraku algoritma izračunavaju se *chroma* značajke pomoću *Short-time* Fourierove transformacije koja je detaljnije objašnjena u poglavljima 3.2.1 i 3.4. Kao ulaz uzimaju se prethodno definirane varijable za veličinu intervala uzorkovanja i učitani audio zapis s njegovom frekvencijom uzorkovanja. Kôd ovog dijela algoritma prikazan je na slici 3.18..

Linija ***Kôd***

```
        chromagram = librosa.feature.chroma_stft(y=y, sr=Fs,
1:      n_fft=window_size, hop_length=hop_length)
```

Sl. 3.18. Kôd primjer za izračun *chroma* značajki

U petom koraku algoritma definira se rječnik akorda. To je zapravo *python* rječnik s vektorima koji opisuju akorde. Vektori su osmišljeni tako da vrijednost *1* predstavlja prisutnost note, a vrijednost *0* nedostatak note u akordu. Note su u vektorima poredane sljedećim redoslijedom: C, C#, D, D#, E, F, F#, G, G#, A, A#, B. U rječniku algoritma nalazi se 24 akorda, a to su akordi navedenih 12 nota u svojoj durskoj i molskoj izvedbi. Radi jednostavnosti prikaza na slici 3.19. su prikazana samo dva vektora C akorda u durskoj i molskoj izvedbi.

Linija ***Kôd***

```
1:      chord_dictionary = {
2:          'C': [1,0,0,0,1,0,0,1,0,0,0,0],
3:          'Cm': [1,0,0,1,0,0,0,1,0,0,0,0],
4-25:   ...
26:     }
```

Sl. 3.19. Kôd primjer za definiranje rječnika akorda

U šestom koraku algoritma pristupa se uspoređivanju dobivenih značajki iz četvrtog koraka s rječnikom akorda definiranog u petom koraku. Uzimaju se redom dobivene *chroma* značajke te se iterira po rječniku kako bi se pronašla najmanja udaljenost odnosno vrijednost koja najmanje odstupa od dobivene. Nakon prolaska kroz cijeli rječnik u listu *recognised_chords* se sprema rezultat koji se najbolje podudara s dobivenim te se uzima iduća značajka i postupak se ponavlja sve dok se ne prođe kroz sve dobivene značajke. Na slici 3.20. prikazan je kôd ovog dijela algoritma.

Linija ***Kôd***

```
1:      recognized_chords = []
2:      for chroma_vector in chromagram.T:
3:          best_match = None
4:          min_distance = float('inf')
5:          for chord_name, chord_template in chord_dictionary.items():
6:              distance = np.linalg.norm(chroma_vector - chord_template)
7:              if distance < min_distance:
8:                  min_distance = distance
9:                  best_match = chord_name
10:         recognized_chords.append(best_match)
```

Sl. 3.20. Kôd primjer za određivanje akorda

U sedmom koraku algoritma kreira se lista u kojoj se nalazi popis akorda s njihovim početnim i krajnjim vremenima. Kroz dobivenu listu *recognised_chords* iz prethodnog koraka se iterira po svakom akordu i vremenskoj oznaci kako bi se dobio predviđeni početak i kraj svakog akorda.

Izlaz iz ovog dijela algoritma je lista naziva *chord_timestamps*. Kôd ovog koraka algoritma vidi se na slici 3.21..

Linija ***Kôd***

```
1:      chord_timestamps = []
2:      current_chord = recognized_chords[0]
3:      current_timestamp = 0
4:
5:      for timestamp, chord in enumerate(recognized_chords):
6:          if chord == current_chord:
7:              continue
            chord_timestamps.append((current_chord, current_timestamp *
8:      hop_length / Fs, timestamp * hop_length / Fs))
9:          current_chord = chord
10:         current_timestamp = timestamp
```

Sl. 3.21. Kôd primjer za izdvajanje početnih vremena akorda

U osmom koraku izvodi se prikaz dobivenih rezultata u 2D prostoru odnosno ravnini. Prikaz se izvodi pomoću biblioteke *Matplotlib* i naredbe *plot*. Iterira se kroz svaki akord iz liste *chord_timestamps* te se crvenom bojom prikazuju predviđeni akordi. Na x-osi prikazano je vrijeme dok se na y-osi nalazi popis svih akorda u rječniku. Izlaz je moguće vidjeti na slici 3.23., a kôd ovog dijela algoritma na slici 3.22..

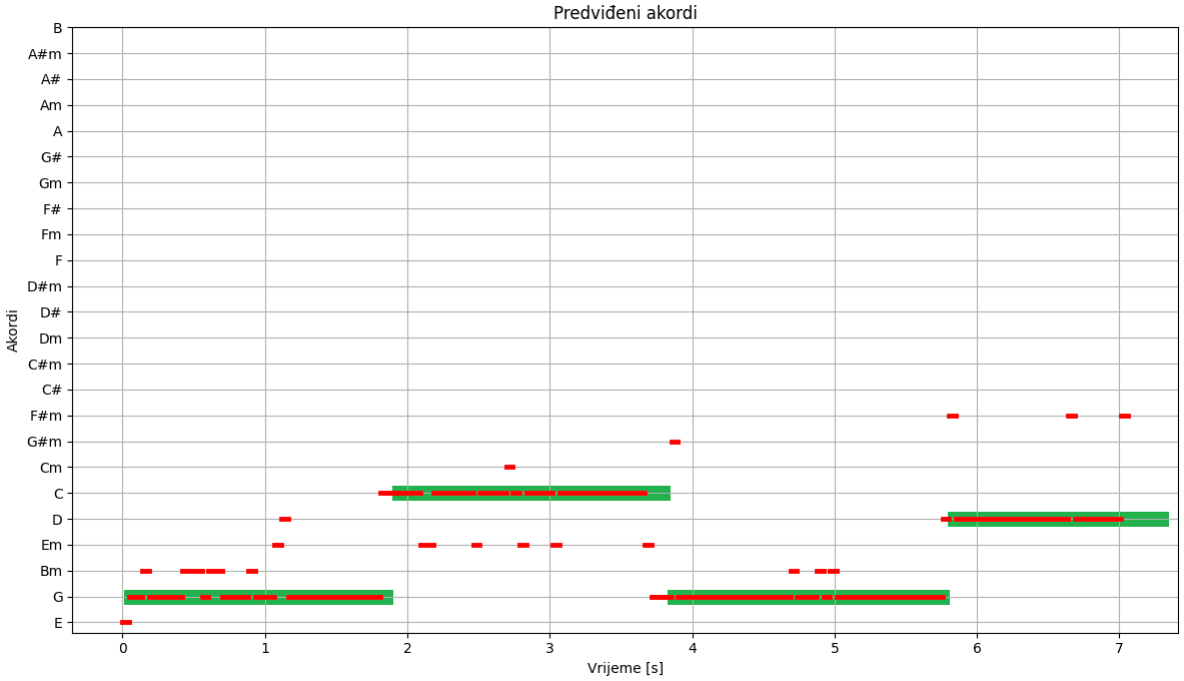
Linija ***Kôd***

```
1:      plt.figure(figsize=(12, 7))
2:
3:      for chord, start, end in chord_timestamps:
4:          plt.plot([start, end], [chord] * 2, color='r', linewidth=3.5)
5:
6:      plt.yticks(list(chord_dictionary.keys()))
7:      plt.xlabel('Vrijeme [s]')
8:      plt.ylabel('Akordi')
9:      plt.title('Predviđeni akordi')
10:     plt.tight_layout()
11:     plt.grid()
12:     plt.show()
```

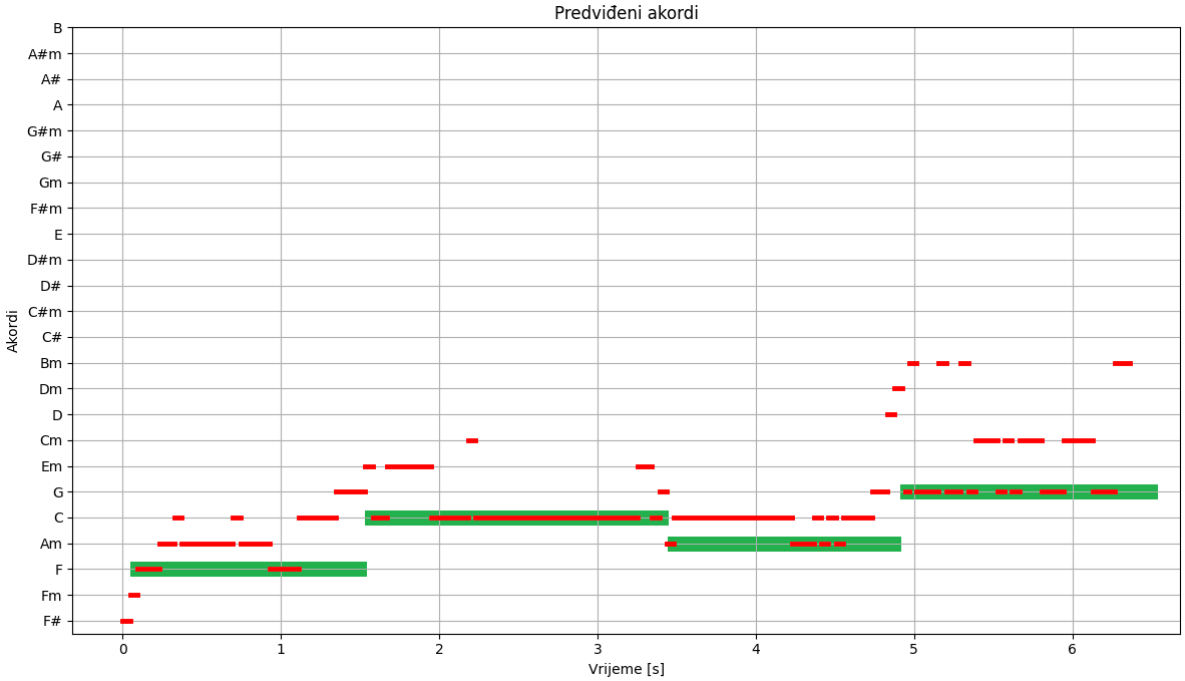
Sl. 3.22. Kôd primjer za prikaz predviđenih akorda

U svrhu testiranja opisanog algoritma provedena je detekcija akorda na dvije melodije koje su izvedene na gitari. Na prikazima crte crvene boje označavaju detektirane akorde dok deblje linije zelene boje označavaju originalne akorde kako i jesu izvedeni u skladbama. U prvom primjeru zvuk instrumenta je podosta jasan i čist dok u drugom postoji prisutnost šuma i distorzije. Analogno tome na prvom primjeru, prikazanom na slici 3.23., može se vidjeti kako je algoritam

podosta dobro predvidio prisutne akorde, dok u drugom primjeru, prikazanom na slici 3.24., postoji dosta nepravilnosti s kojima algoritam nije znao pravilno postupati te je zato i krivo detektirao akorde. U navedenim primjerima se može vidjeti kako čak i male nepravilnosti u zvuku mogu značajno utjecati na proces detekcije akorda. Stoga je podosta teško stvoriti robustan sustav



Sl. 3.23. Prikaz detekcije akorda melodije s čistim zvukom



Sl. 3.24. Prikaz detekcije akorda melodije s prisutnim šumom i distorzijom

kako za problem *AMT*-a tako i za problem detekcije akorda jer da bi sustav mogao dobro generalizirati treba dobro opisati odnose između harmonijskih značajki unutar melodije. Takve značajke često su teško vidljive i prepoznatljive u računalnom prikazu *audio* zapisa te se zato često koristi pristup strojnog i dubokog učenja za ovakve tipove problema.

4. Izvlačenje značajki glazbe

Kako bi se dobro odradila transkripcija glazbe, potrebno je imati potpune i kvalitetne podatke odnosno značajke prethodno izdvojene iz glazbe. Automatsko određivanje transkripcije, bilo ono provođeno jednostavnim načinima ili pomoću naprednih načina, kao što su neuronske mreže, zahtijeva unaprijed određene strukturne značajke glazbe odnosno skladbe koju se transkriptira. Neke od osnovnih značajki, kao što su visina tona, polifonija i harmonija, su već prethodno određene i objašnjene u prošlom poglavlju (3). Fokus ovog poglavlja bit će izvlačenje odnosno određivanje naprednijih strukturnih značajki glazbe, kao što su počeci nota, ritam, tempo i takt.

4.1. Određivanje početka note

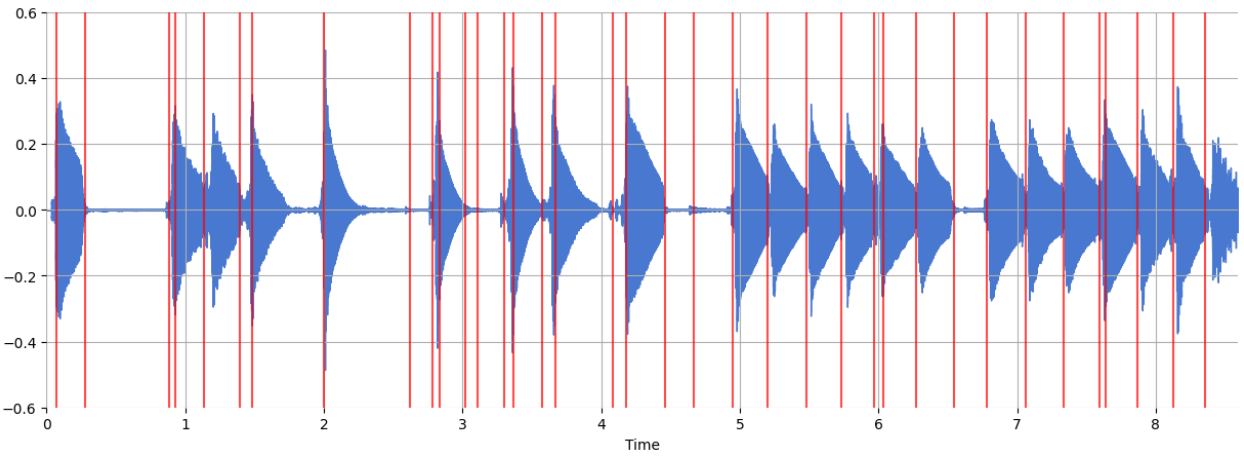
Jedni od najbitnijih značajki potrebnih za automatsku transkripciju glazbe su počeci nota. Njihovo određivanje ponekad može biti podosta zahtjevno i nejasno. Kako bi se kvalitetno odredili počeci nota potrebno je promatrati nagle promjene odnosno konkretnije poraste energije koji se događaju na istima [5, str. 311-322]. U praksi želi se pronaći interval odsviranog tona u fazi napada, odnosno u fazi gdje se ton gradi i dostiže svoju maksimalnu amplitudu to jest intenzitet. Faza napada predstavlja prvi dio *ADSR* modela detaljnije objašnjenog u poglavlju 2.2.4. Jedan od problema određivanja početaka nota je taj što početak može biti određen i promjenjivom fazom, a ne samo fazom napada. Promjenjivu fazu teže je analizirati jer ona predstavlja zvučnu komponentu kratkog trajanja i visoke amplitude koja se može pojaviti ili na početku glazbenog tona ili prilikom nekog većeg glazbenog događaja. U fazi otpuštanja tona također može doći do pojave promjenjive faze zvuka. U navedenim promjenjivim fazama zvuka signal evoluira brzo i na nepredvidiv i kaotičan način. Još jedan od problema određivanja početaka nota je korištenje instrumenata koji nemaju točno naglašene početke i krajeve nota već se zvuk na njima izvodi fluidno, kao što je to na primjer violina. U stvarnosti ne postoji točno vrijeme početka note odnosno konstanta na vremenskom pravcu za koji se može reći da je početak već postoji period odnosno interval (faza napada ili promjenjiva faza) koji se asocira i za koji kažemo da predstavlja početak note. Generalna ideja je uhvatiti nagle promjene u zvuku koje često predstavljaju početak promjenjivih faza, dok se za note koje počinju fazom napada uzima vrijeme u kojem amplituda signala naglo započinje rasti. Neke od metoda odnosno značajki koje pomažu u detekciji početaka nota su: pristup analizom energije (engl. *Energy-Based Novelty*), pristup analiziranjem spektra (engl. *Spectral-Based Novelty*), pristup analiziranja faze zvuka (engl. *Phase-Based Novelty*) i pristup korištenjem kompleksne domene (engl. *Complex-Domain Novelty*). U pristupu analiziranja energije traže se nagla povećanja energije signala koja se direktno koreliraju s udarcem bata po žici klavira, udarcem po

žici gitare, udarcem o opnu bubnja i slično. Pristup analiziranja spektra bazira se na detekciji promjena frekvencija jer one bolje opisuju početke nota u polifonijskoj glazbi zbog same činjenice da razni instrumenti proizvode zvuk u različitim frekvencijskim intervalima. Pristup analiziranja faze zvuka osim magnitude spektralnih koeficijenata uzima u obzir i informaciju o fazama i faznim pomacima signala. Glavna ideja je ta da stacionarni tonovi imaju stabilnu fazu, dok tonovi s promjenjivim fazama nemaju stabilnu fazu. Pristup korištenjem kompleksne domene pripada robusnijim načinima detekcije te za svoj rad koristi i informaciju o fazi signala i informaciju o magnitudi, a koristi se kada je koeficijent magnitude dovoljno malen da uzrokuje kaotično ponašanje u stabilnim dijelovima signala. Kao primjer naveden je kôd na slici 4.1. koji koristi već korištenu biblioteku *librosa* za određivanje početaka nota. Korištena naredba *onset_detect* primjenjuje pristup analize spektra za detekciju početaka nota. Na slici 4.2. može se vidjeti izlaz navedenog kôda te se može uočiti kako nisu svi počeci pravilno određeni, već postoje dodatne detekcije koje tamo ustvari ne bi trebale biti. Počeci su karakterizirani vertikalnim zadebljanjima, a detektirani počeci su označeni crvenim crtama.

Linija ***Kôd***

```
1:      y, sr = librosa.load('test.wav')
        onset_frames = librosa.onset.onset_detect(y=y, sr=sr, wait=1,
2:      pre_avg=1, post_avg=1, pre_max=1, post_max=1)
3:      onset_times = librosa.frames_to_time(onset_frames)
4:      plt.figure(figsize=(15, 5))
5:      librosa.display.waveshow(y=y, sr=sr)
6:      plt.vlines(onset_times, -0.6, 0.6, color='r', alpha=0.7)
```

Sl. 4.1. Kôd primjer za detekciju početaka nota



Sl. 4.2. Prikaz detekcije početaka nota

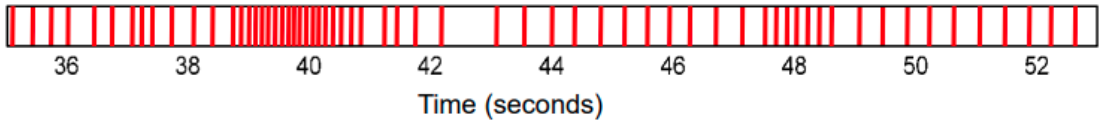
4.1.1. Određivanje završetka note

Problem kod određivanja završetaka nota je taj što završeci često mogu biti dvosmisleni te se zato često ne pridodaje veliki značaj detektiranima prilikom vrednovanja istih [9, str. 4], odnosno u obzir se uzima tolerancija na pogrešku. Nerijetko se događa da se ni glazbenici ne mogu uskladiti oko samih završetaka nota unutar određenih skladbi. Problem se najčešće rješava tako da se skladba izvodi intuitivno to jest po osjećaju se određuju završetci odnosno trajanje nota. Problem u određivanju predstavlja i preklapanje samih nota jer često se događa da iduća nota kreće s izvedbom tijekom trajanja trenutačne. U procesu *AMT*-a obično se za rješenje odabire ili kraj faze otpuštanja tona (objašnjeno u poglavlju 2.2.4) odnosno uzima se trenutak kada se ton note stopi sa sveukupnim šumom u skladbi ili se uzima trenutak netom prije početka sljedeće note.

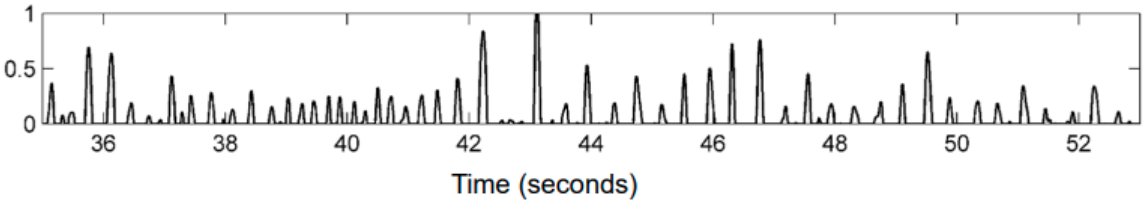
4.2. Određivanje ritma

Ritam se u glazbenom smislu može definirati kao jedna od glavnih značajki glazbe koja opisuje odnos između zvukova različitih jačina i trajanja odnosno opisuje vremenski raspored događaja [19]. Ritam predstavlja vodoravni aspekt glazbe jer opisuje glazbene značajke i strukture koje de događaju u određenom intervalu. Mjerna jedinica za određivanje ritma je takt. Takt se u notnom zapisu prikazuje na početku i ima zapis oblika razlomka. Takt odnosno ritam je često definiran kao brzina lupkanja nogom koju slušatelji neke skladbe izvode. Takt se često može odrediti promatranjem odnosno praćenjem udaračkih instrumenata unutar skladbe, ukoliko skladba takve posjeduje. Problem kod takta je taj što, unutar jedne skladbe, on može biti promjenjiv te su zbog toga potrebne robusne metode koje su sposobne opisati lokalne mjere ritma. U procesu *AMT*-a takve metode se često oslanjaju na detektirane početke nota kako bi odredili takt [5, str. 334-351]. U suštini proces određivanja takta odvija se tako što se *Short-time* Fourierovom transformacijom odrede počeci nota (kako je opisano u poglavlju 4.1) na način da se nad dobivenim frekvencijama primjeni logaritamska funkcija kako bi se bolje naglasile slabije komponente dobivenog spektra. Zatim se računa diskretna derivacija nad frekvencijskim pojasevima kako bi se odredili energetski skokovi odnosno povećanja koja se onda sumiraju u odnosu na vrijeme. Time se dobiva funkcija značajki odnosno noviteta (engl. *novelty function*) koja predstavlja nagle promjene su spektru u odnosu na vrijeme. Prema navedenoj funkciji značajki računaju se lokalni prosjeci koji se onda oduzimaju od same funkcije značajki kako bi se prikazali trenutci koji predstavljaju novine unutar skladbe odnosno u većini slučajeva početke nota. Nad dobivenim izlazom računaju se funkcije prevladavajućeg lokalnog pulsa (engl. *predominant local pulse, PLP*) kako bi se dobile vremenske konstante koje opisuju takt. Primjer navedenih koraka ovog postupka vidi se na sljedećim slikama,

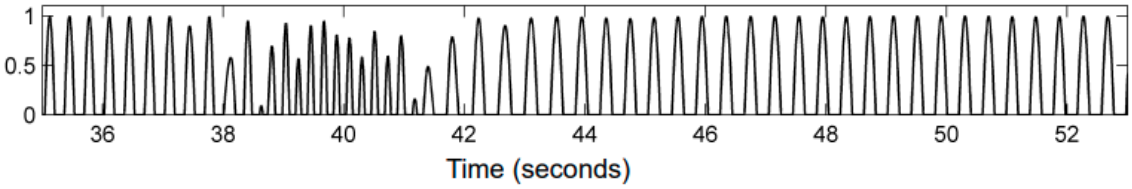
gdje su na slici 4.3. prikazani određeni počeci nota, na slici 4.4. izračunata funkcija noviteta te na slici 4.5. izračunate *PLP* funkcije čiji vrhovi prikazuju takt koji je u ovom slučaju promjenjiv.



Sl. 4.3. Prikaz početaka nota



Sl. 4.4. Prikaz izračunatih funkcija noviteta



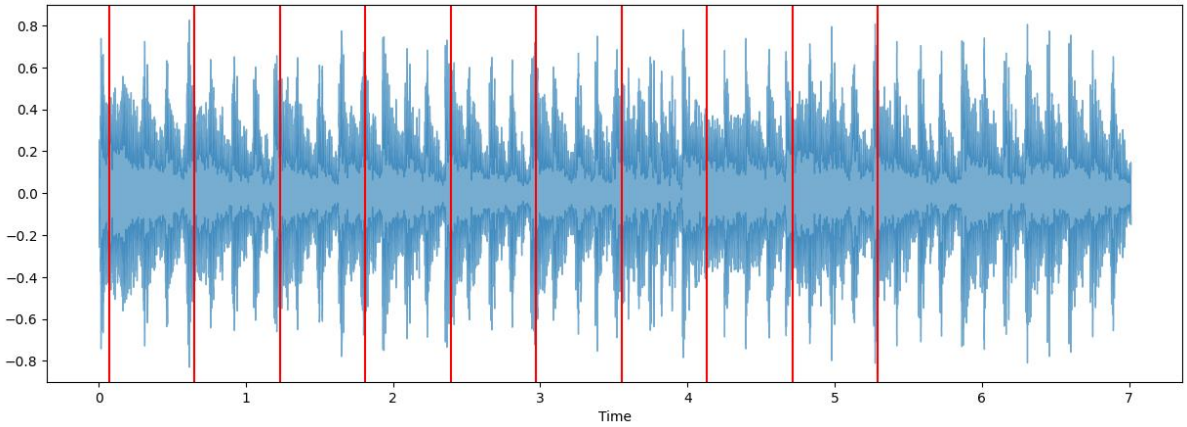
Sl. 4.5. Prikaz prikaz izračunatih *PLP* funkcija

Sve navedeno također je moguće izvršiti programskim putem pomoću već korištene biblioteke *librosa*. Biblioteka sadrži naredbu naziva *beat_track* koja za zadani ulaz daje izlaz u obliku vremenskih točaka koje je moguće prikazati zajedno s valnim oblikom audio zapisa. Na slici 4.5. prikazan je kôd za računanje i prikaz takta, a na slici 4.6. su okomitim crvenim crtama označeni trenutci lupkanja takta. U ovom primjeru izraženi su udarci u doboš bubanj koji u stvarnosti i opisuju pravi takt te ih je program skoro u cijelosti uspio točno odrediti.

Linija ***Kôd***

```
1:      y, sr = librosa.load('painkiller.wav')
        tempo, beat_times = librosa.beat.beat_track(y=y, sr=sr,
2:      start_bpm=60, units='time')
3:      plt.figure(figsize=(15, 5))
4:      librosa.display.waveshow(y=y, sr=sr, alpha=0.6)
5:      plt.vlines(beat_times, -0.9, 0.9, color='r')
6:      plt.ylim(-0.9, 0.9)
```

Sl. 4.5. Kôd primjer za određivanje takta



Sl. 4.6. Prikaz određivanja takta

4.3. Određivanje tempa

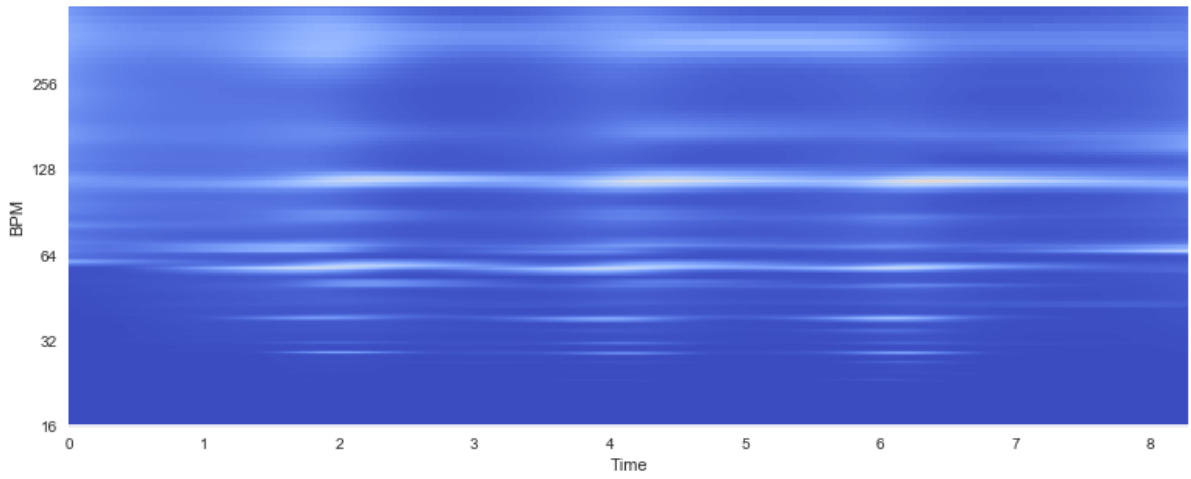
U glazbenoj terminologiji, tempo predstavlja brzinu izvođenja glazbe [20]. Tempo se označava u otkucajima po minuti (engl. *beats per minute*) odnosno koliko udaraca takta je prisutno u minuti. Tempo skladbi daje doživljaj, na primjer ukoliko je tempo veći broj kao 150 otkucaja u minuti, skladba će zvučati življe. Također se označava i talijanskim nazivima kao što su: *allegro* (brzo), *adagio* (sporo), *moderato* (srednje brzo), itd. Problem kod određivanja tempa za proces *AMT*-a je taj što je on dosta često promjenjiv unutar skladbe. Da bi se izvršilo određivanje tempa potrebno je zadnji dio postupka za određivanje takta (objašnjenom u poglavlju 4.2), u kojemu se računaju *PLP* funkcije, ponoviti na svim lokalnim dijelovima skladbe. Time se dobiva graf *PLP* funkcija koji se također može prikazati kao tempogram (posebna vrsta grafa koji prikazuje tempo u ovisnosti o vremenu). Kao primjer, na slici 4.8. prikazan je tempogram koji predstavlja izlaz kôda sa slike 4.7. na kojem je isti pomoću naredbe *tempogram* iz biblioteke *librosa* izračunat. Na tempogramu je moguće vidjeti linije gustoće bijele boje koje opisuju tempo u određenom vremenskom periodu. U ovom primjeru tempo je podosta konstantan te iznosi otprilike 120

Linija ***Kôd***

```
1:      y, sr = librosa.load('test.wav')
2:      hop_length = 200
        onset_env = librosa.onset.onset_strength(y=y, sr=sr,
3:      hop_length=hop_length, n_fft=2048)
        tempogram = librosa.feature.tempogram(onset_envelope=onset_env,
4:      sr=sr, hop_length=hop_length, win_length=400)
        librosa.display.specshow(tempogram, sr=sr, hop_length=hop_length,
5:      x_axis='time', y_axis='tempo')
```

Sl. 4.7. Kôd primjer za određivanje tempa

otkucaja u minuti. Globalni tempo, koji nije uvijek najtočnija reprezentacija, može se dobiti naredbom *beat_track* koja je prikazana na slici 4.5..



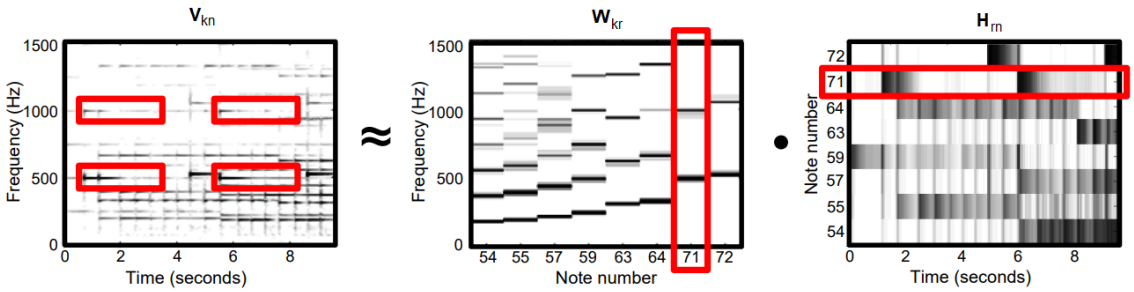
Sl. 4.8. Prikaz određivanja tempa

5. Napredna automatska transkripcija glazbe

Razvojem tehnologija i računalnih resursa proces automatske transkripcije glazbe pribjegao je implementaciji naprednijih i zahtjevnijih metoda. Takve metode uvelike uključuju uporabu strojnog učenja i neuronskih mreža te su se pokazale bolje u performansama transkripcije i generaliziranju različitih instrumenata od standardnih konvencionalnih metoda. Osim strojnog učenja istraživačima ovog područja nametnule su se još neke metode, kao što su množenje nenegativnih matrica, kao jedno od optimalnih i dobrih rješenja. Unutar ovog poglavlja bit će opisan dio naprednih metoda za proces AMT-a zajedno s jednostavnim praktičnim primjerima tih metoda i njihovim izlazima.

5.1. Množenje nenegativnih matrica

Množenje nenegativnih matrica (engl. *nonnegative matrix factorization, NMF*) pripada naprednijim metodama za proces *AMT*-a, a cilj te metode je ustvari priprema podataka za daljnju obradu ili čak uređivanje. *NMF* proces ustvari dekompozira navedenu skladbu, ali ne u smislu razdvajanja instrumenata i glasova već u smislu razdvajanja signala u notne audio događaje, gdje je svaki događaj direktno koreliran s pojedinom notom u glazbenom zapisu [5, str. 456-478]. Takvim načinom obrade signala otvaraju se nove mogućnosti uređivanja i manipuliranja audio signala. Glavna zamisao cijelog postupka je imati matricu **V** s realnim koeficijentima koji su veći ili jednaki od 0, dimenzija K x N, koja je produkt faktorizacije dviju matrica **W** i **H**. Matrice **W** i **H** također imaju nenegativne koeficijente te su im dimenzije K x R za matricu **W** i R x N za matricu **H**. Parametar R se često smatra rangom faktorizacije i izabire se takav da je manji od parametara K i N. Matrica **V** nije nužno točan produkt faktorizacije već je aproksimacija faktorizacije odnosno vrijedi $\boldsymbol{V} \approx \boldsymbol{WH}$. Računski se ovaj problem rješava minimiziranjem udaljenosti (Euklidske) između matrice **V** i produkta faktorizacije **WH** s time da se mora poštovati ograničenje nenegativnosti. Minimiziranje se često izvršava metodom gradijentnog spusta. Matrica **V** u procesu *AMT*-a predstavlja spektrogram magnituda frekvencija u ovisnosti o vremenu zadanog audio zapisa. Matrica **W** se također naziva i rječnik ili matrica šablona u koju je cilj spremiti u obliku stupaca, što manji broj vektora šablona koji predstavljaju frekvenciju pojedinih nota. Matrica **H** se naziva matrica aktivacije te se u njezine redove spremaju vektori koji govore o tome kada u skladbi dolazi do pojave pojedinog vektora šablone. Navedeno je moguće vidjeti na slici 5.1.. *NMF* metoda dominantna je metoda u procesu *AMT*-a jer je računski efikasna i zbog toga što dodaje poboljšanja u analitičkim postupcima. Na primjer, nekad je podosta teško izvući korisne informacije iz spektrograma magnituda frekvencija, ali je iz istog rastavljenog na dvije



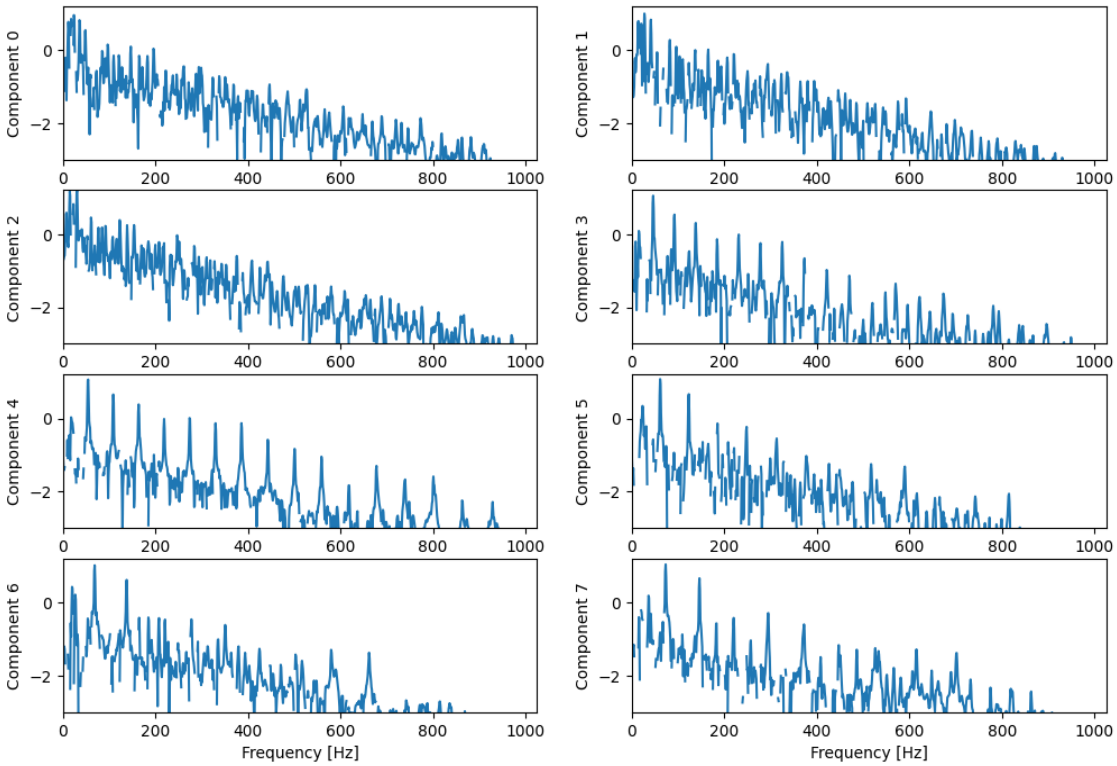
Sl. 5.1. Množenje nenegativnih matrica

matrice puno lakše prepoznati i interpretirati razne glazbene strukture. Pomoću biblioteke *librosa* moguće je odraditi navedeni postupak faktorizacije korištenjem naredbe *decompose*. Proces funkcionira na način da se odredi *Short-time* Fourierova transformacija nad učitanim *audio* zapisom nakon čega se dobiveni spektrogram rastavlja na komponente magnitude i faze te se komponenta magnitude prosljeđuje naredbi *decompose* koja obavlja *NMF* proces i kao izlaz daje matrice **W** i **H**. Na slici 5.2. prikazan je kôd navedenog procesa, dok su na slikama 5.3. i 5.4. prikazani vektori odnosno komponente marice **W** i **H**. Zbog jednostavnosti prikazan je samo dio kôda za ispis matrice **W** jer je za ispis **H** matrice kôd identičan. Za količinu komponenti odabran je broj 8. Struktura grafova (oscilirajući i blago padajući) na spektralnom prikazu odnosno na prikazu komponenti matrice **W** je takva zbog prisustva harmonika svake note, ustvari točna frekvencija je samo ona na prvoj oscilaciji s lijeve strane grafa.

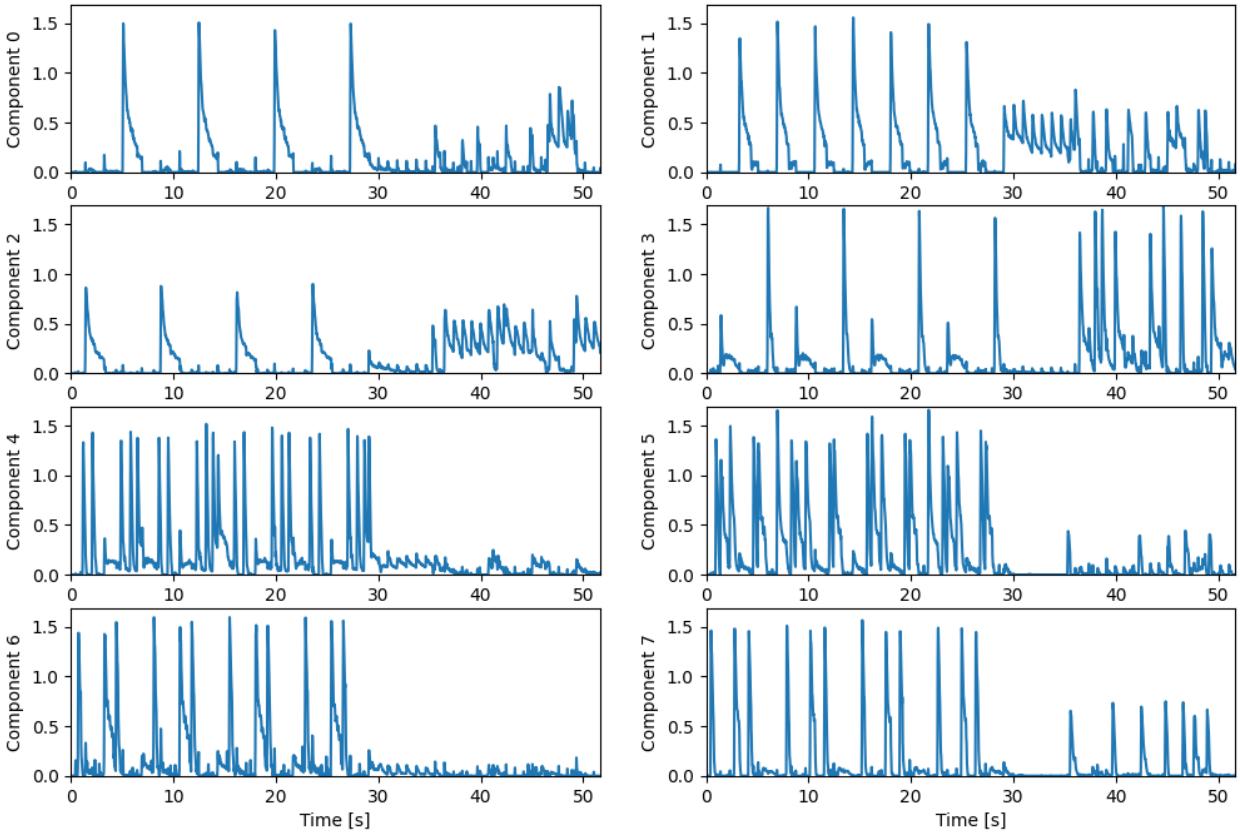
Linija ***Kôd***

```
1:      y, sr = librosa.load('test.wav')
2:      S = librosa.stft(y)
3:      Y, Y_phase = librosa.magphase(S)
4:      n_components=8
        W, H = librosa.decompose.decompose(Y, n_components=n_components,
5:      sort=True)
6:      plt.figure(figsize=(12, 8))
7:      logW = numpy.log10(W)
8:      for n in range(n_components):
9:          plt.subplot(int(numpy.ceil(n_components/2)), 2, n+1)
10:         plt.plot(logW[:,n])
11:         plt.ylim(-3, logW.max())
12:         plt.xlim(0, W.shape[0])
13:         plt.ylabel('Component %d' % n)
14:         plt.xlabel('Frequency [Hz]')
```

Sl. 5.2. Kôd primjer za *NMF* proces



Sl. 5.3. Prikaz komponenti matrice **W**

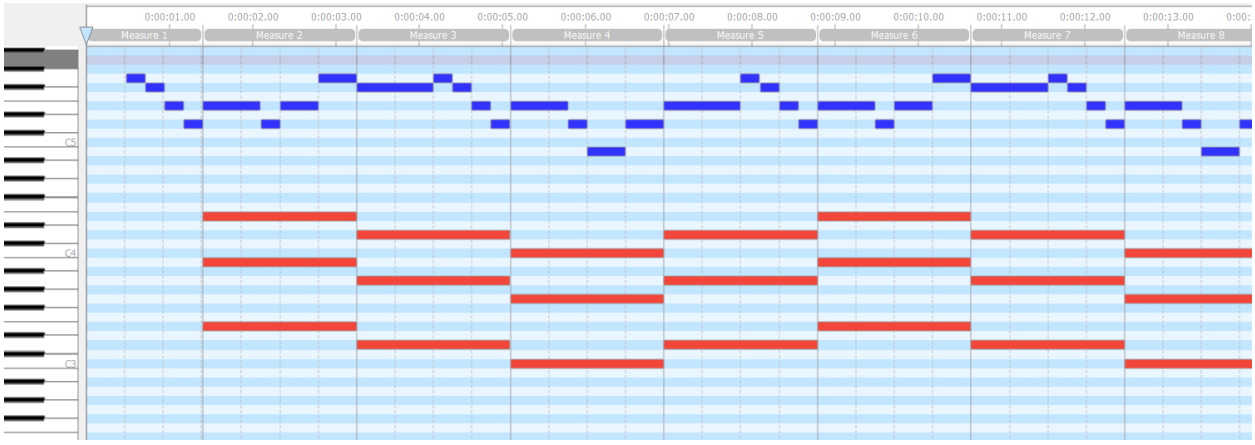


Sl. 5.4. Prikaz komponenti matrice **H**

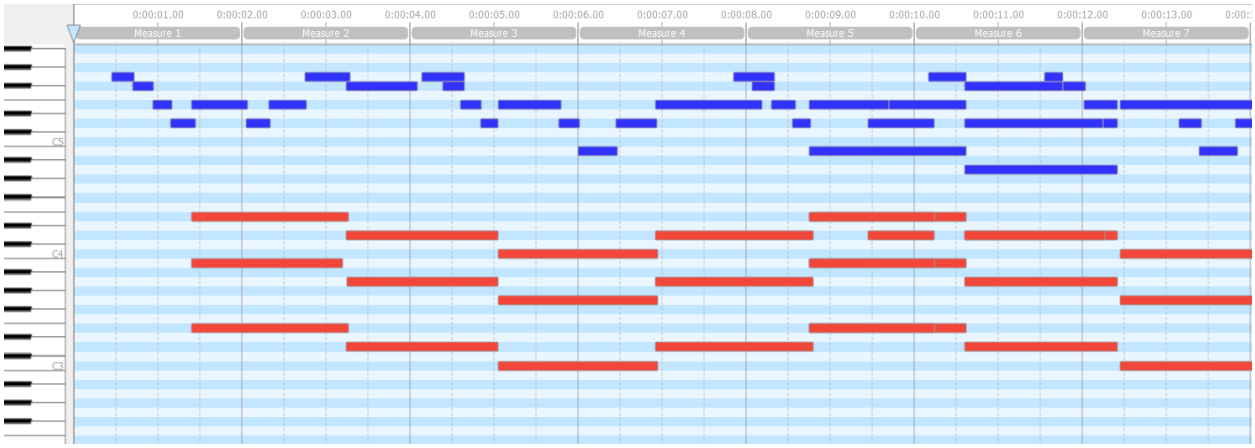
5.2. Neuronske mreže

Umjetna inteligencija (engl. *artificial intelligence, AI*) odnosi se na sposobnost računala da izvode zadatke koji se inače rješavaju ljudskom inteligencijom kao što su to na primjer: prepoznavanje govora, učenje, donošenje odluka i rješavanje problema [21]. U sklopu područja umjetne inteligencije kao jedno od rješenja navedenih zadataka nameću se neuronske mreže (engl. *neural networks, NN*). Neuronske mreže predstavljaju modele strojnog učenja koji su stvoreni na principima neuronskih organizacija prisutnim u stvarnom svijetu, kao što je to na primjer ljudski mozak [22]. Neuronske mreže sastavljene su od čvorova zvanih umjetni neuroni koji služe za procesiranje zadanog ulaza, a spojeni su međusobno težinama koje se prilagođavaju tijekom treniranja. Treniranje ili učenje mreže odnosi se na kalibriranje mreže stvarnim primjerima to jest pokušava se izgraditi model koji može opisati strukturu i funkcionalnu povezanost zadanih ulaznih podataka. Neuronske mreže se u zadnje vrijeme puno upotrebljavaju u svrhu procesa *AMT*-a jer su stanju naučiti nelinearnu funkciju (ili kompoziciju funkcija) koja opisuje glazbene strukture unutar skladbi [9, str. 6-7]. Neuronske mreže također bolje generaliziraju različite instrumente od standardnih metoda no ipak postoje i specijalno trenirane mreže za određeni instrument. Za problem *AMT*-a istaknula se uporaba *LSTM* slojeva (engl. *Long Short-Term Memory*) čiji su benefiti bolje dohvaćanje i analiza spektralnih karakteristika nota koje evoluiraju kroz vrijeme te također bolja obrada glazbenih struktura koje se događaju između samih nota unutar skladbe. Na primjer, nakon odsvirane 2 ili tri note modeli mogu lakše zaključiti, pa čak i predvidjeti, koja će iduća nota biti odsvirana. Također u neuronskim mrežama moguće je propagirati signal unazad zbog ponovne i detaljnije obrade ili spajati više mreža u jednu kako bi se poboljšale performanse. Na primjer, *Google Brain's Onset and Frames Network* je mreža koja se sastoji od dvije manje mreže od kojih prva služi za predviđanje početaka nota te se rezultati prosljeđuju kao ulaz za mrežu koja predviđa duljinu trajanja istih. Kao demonstrativni primjer u svrhu ovog rada odabran je model *MT3* (engl. *Multi-Task Multitrack Music Transcription*) koji je nastao od strane *Magenta* projekta koji se bazira na istraživanju strojnog učenja kao alata za kreativne procese [23]. Model *MT3* baziran je na transformatorima koji predstavljaju poseban tip modela unutar područja strojnog učenja koji je specijalizirano dizajniran za obradu sekvencijalnih odnosno slijednih podataka kao što su to na primjer tekst ili nizovi znakova. Također je korišteno i *AnthemScore* softversko rješenje tvrtke *Lunaversus* koje se bazira na korištenju umjetne inteligencije za automatsku transkripciju glazbe [24]. Rješenje koristi metode strojnog učenja kako bi detektiralo note i posložilo ih u notne segmente. Na slici 5.5. prikazane su note klavirskog dijela prvih 14 sekundi pjesme *Empire of the Clouds* grupe *Iron Maiden* u vizualiziranom *MIDI* formatu. Na slici

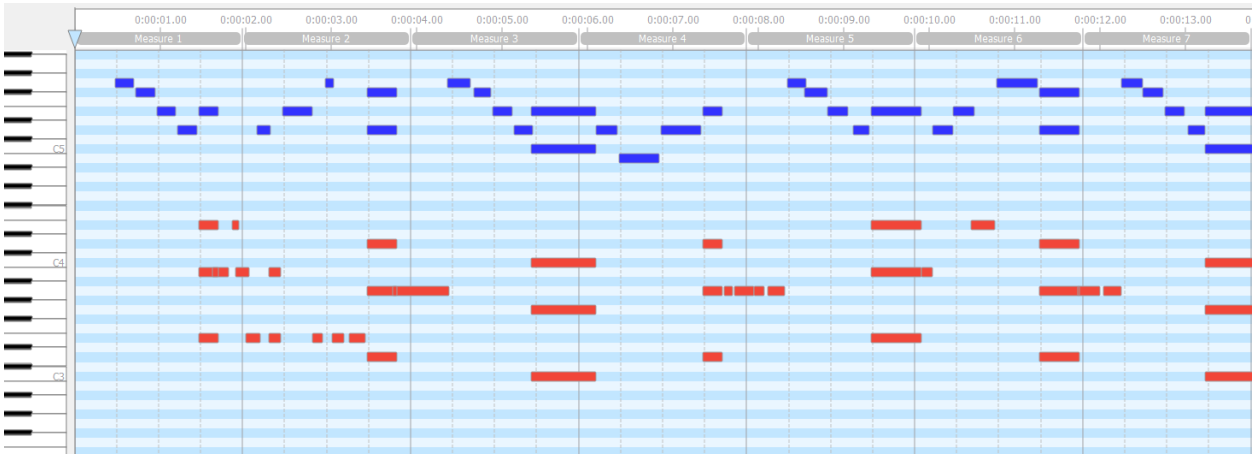
5.6. prikazane su note određene *MT3* modelom dok su na slici 5.7. prikazane note određene softverom *AnthemScore*. Iz slika se zaključuje kako *MT3* model u većini slučajeva dobro određuje trajanje nota no često zna predvidjeti viška note dok softver *AnthemScore* ima poteškoća s određivanjem trajanja nota no same visine tonova određuje bolje nego model *MT3*.



Sl. 5.5. Originalne note pjesme *Empire of the Clouds*



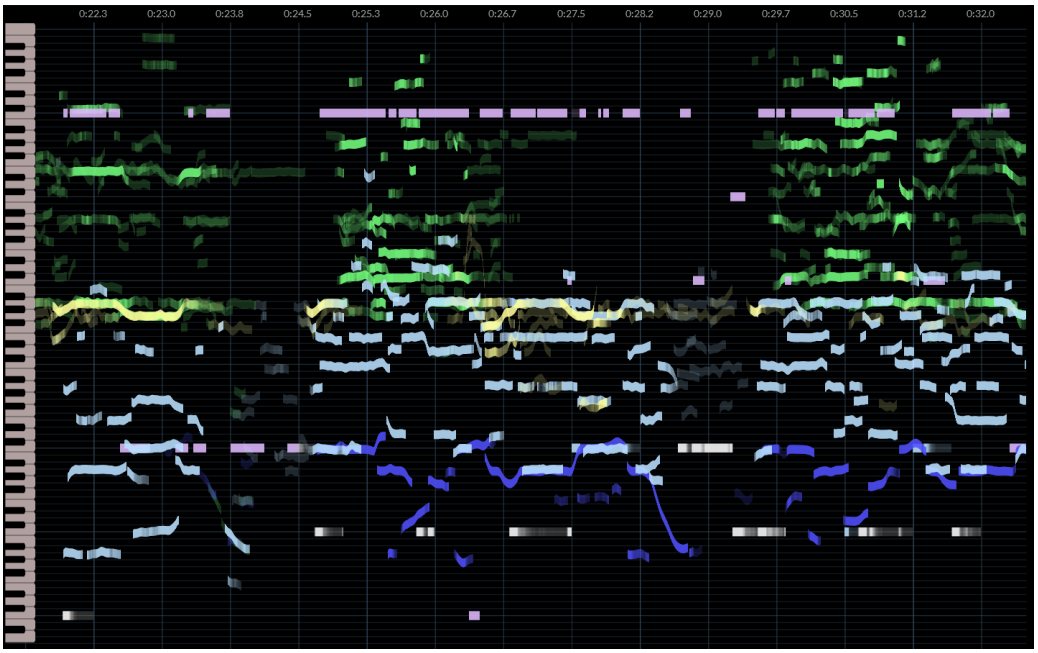
Sl. 5.6. Note određene *MT3* modelom



Sl. 5.7. Note određene *AnthemScore* softverom

5.3. Razdvajanje instrumenata unutar glazbe

Unutar područja koje se bavi obradom i izvlačenjem značajki iz glazbe osim problema *AMT*-a pripada i problem razdvajanja izvora odnosno instrumenata iz glazbenom zapisa (engl. *audio/music source separation*). Iako se navedena dva problema tretiraju odvojeno, zapravo su međusobno jako povezani po principima to jest metodama za obradu i izvlačenje značajki iz glazbe. Naime problem razdvajanja instrumenata može uvelike pridonijeti procesu *AMT*-a i u dosta slučaja postojeća rješenja za oba problema se integriraju u jedan sustav. Primjer takve integracije je navedeni *MT3* model iz poglavlja 5.2, koji osim određivanja nota odrađuje i razdvajanje samih instrumenata. Proces razdvajanja instrumenata većinom se koristi u procesu *AMT*-a kao predobrada glazbenog zapisa jer ustvari samim razdvajanjem se polifonijska glazba svodi na individualne kanale monofonijske glazbe u kojoj svaki kanal predstavlja jedan instrument ili glas. Ovakvim načinom olakšava se proces *AMT*-a jer puno je lakše i preciznije odrediti note kada se obrađuje *audio* zapis s jednim instrumentom nego zapis u kojemu se preklapa više instrumenata u istom vremenskom periodu. Sama softverska rješenja razdvajanja instrumenata iz glazbe se za svoj rad uvelike oslanjaju na metode koje koriste informacije kao što su boja zvuka (objašnjeno u poglavlju 2.2.4) i frekvencijski raspon instrumenata. Većina takvih rješenja implementirano je kao neuronske mreže. Uz prednosti koje ovaj proces donosi procesu *AMT*-a postoje i nedostatci, kao što je to na primjer razvijanje posebnih neuronskih mreža kako bi se kvalitetno razdvojili instrumenti kao i istraživanje novog područja (koje kao i područje *AMT*-a nije potpuno usavršeno) obrade glazbenog signala što predstavlja veliki vremenski i resursni trošak. Kao primjer za ovakva rješenja na slici 5.8. prikazan je izlaz programa *RipX* tvrtke *Hit'n'Mix* na kojem je svaki zasebni instrument i glas razdvojen i naznačen posebnom bojom te tako na primjer svijetlo plava boja predstavlja zvuk gitare, žuta predstavlja vokal pjevača, plava predstavlja bas, roza udaraljke itd. Kao još jedno od kvalitetnih rješenja izdvaja se i sustav *Stems 2.0* tvrtke *Atomix Productions*, koji je ukomponiran u softver zvan *VirtualDJ*. Sustav je u mogućnosti razdvajati izvore glazbe u stvarnom vremenu i kao i *RipX* softver uvelike je baziran na umjetnoj inteligenciji.



Sl. 5.8. Razdvajanje izvora glazbe pomoću softvera *RipX*

6. Budući zadaci i problemi AMT-a

Postupak automatske transkripcije glazbe predstavlja široko područje koje je gotovo nemoguće svesti samo na jedan zadatak već mora biti rascjepkano na više manjih individualnih zadataka. U svakom od tih zadataka postoje specifični problemi na kojima treba na poseban način pristupiti i stvoriti nove metode za rješavanje navedenih. U ovom poglavlju bit će objašnjeni neki od trenutačno glavnih ciljeva i problema koji zahtijevaju precizno istraživanje kako bi pridonijeli procesu *AMT*-a.

6.1. Music language models

Kao što je navedeno u prethodnom poglavlju (5.3), problem *AMT*-a je povezan s mnogo ostalih problema u području računalne obrade glazbe. Jedan od takvih problema koji može pridonijeti postupku *AMT*-a je problem automatskog prepoznavanja govora (engl. *Automatic Speech Recognition, ASR*) [9, str. 8]. Kako se *ASR* sustav sastoji od akustičnih odnosno audio komponenti i jezičnih komponenti tako se i *AMT* sustav sastoji od audio komponenti, pozadinskih notnih struktura i glazbenih struktura. Zato *AMT* sustavi integriraju glazbene jezične modele (engl. *music language models, MLM*), korištene u *ASR* procesima, koji modeliraju razne notne strukture i visoko dimenzionalne veze u serijama noti u polifonijskom kontekstu s ciljem poboljšanja performansi transkripcije. Iako su se ovakvi sustavi na prvi dojam pokazali kao dobri s obećavajućim rezultatima i dalje se ovakav način automatske transkripcije mora još podosta razvijati kako bi se postigle visoke performanse.

6.2. Problem raznolikosti glazbala

U trenutačnom uspjehu procesa *AMT*-a još se uvijek nije pronašlo optimalno generalno rješenje. Jedan od problema tog „neuspjeha“ je problem raznolikosti glazbala odnosno instrumenata. Naime, na svijetu postoji široki asortiman različitih vrsta i izvedbi instrumenata, a kako se neuronske mreže i duboko učenje nameću kao najpogodnije rješenje problema *AMT*-a, gotovo je nemoguće naučiti neuronsku mrežu i stvoriti model koji generalizira postupak transkripcije za sve postojeće instrumente. U sklopu istraživanja i donošenja rješenja problema *AMT*-a, često se za učenje mreža koristi jedan ili nekolicina instrumenata koji skupa idu u neki veliki skup pjesama (orkestralni instrumenti kao što su violina i truba idu zajedno, električna gitara i bubnjevi idu zajedno, tamburica i kontrabas idu zajedno, ...). Modeli trenirani samo na raspoznavanju nota jednog instrumenta često imaju podosta velike uspjehe u automatskoj transkripciji glazbe ukoliko se validacija izvršava na pjesmama s tim specifičnim instrumentom. Potpuno generalna rješenja

možda nikada neće biti moguća, ali daljnjim razvijanjem ovakvih sustava moći će se kvalitetno transkriptirati velika većina popularne glazbe.

6.3. Problem raznolikosti žanrova

Još jedan od problema AMT-a koji je ujedino vezan za prethodno navedeni problem u poglavlju 6.2 je problem raznovrsnosti žanrova glazbe. Kao što kod problema modeliranja modela transkripcije za specifični instrument rezultati često opadaju čim se od modela traži raspoznavanje nekog drugog instrumenta, tako će i prilikom transkripcije pjesme nekog žanra, na kojem model nikad nije imao priliku učiti, rezultati značajno opadati. Ova dva navedena problema uvelike otežavaju izgradnju generalnog modela transkripcije jer uvode previše značajki i veza o kojima model treba učiti. Neka od rješenja predlažu integriranje rješenja automatskog prepoznavanja žanrova i sustava za identifikaciju instrumenata u sustave za automatsku transkripciju glazbe kako bi se prilagodili parametri takvih sustava i kako bi se generirale transkripcije za specifične žanrove [25]. Ovakvi sustavi su još uvijek ili u fazi istraživanja ili u fazi rane implementacije te nije moguće napraviti procjenu o tome kada će takvi sustavi biti dostupni i također kakve će njihove performanse biti.

6.4. Evaluacijske metrike

Većina rješenja problema *AMT*-a vrednovana su korištenjem metrika danim setom podataka *MIREX Multiple-F0 Estimation and Note Tracking public evaluation tasks* [4]. Unutar tog seta podataka dana su tri tipa metrika za evaluiranje takvih sustava, a to su *frame-based, note-based* i *stream-based* metrike koje se direktno koreliraju s razinama transkripcije glazbe [9, str. 9]. Iako navedeni setovi metrika imaju svoje pogodnosti, moguće je dovesti u pitanje ako te metrike odgovaraju ljudskoj percepciji točnosti transkripcije glazbe. Na primjer, nota koja je predviđena u pjesmi, a koja ne postoji u stvarnom audio zapisu bi se moglo klasificirati kao značajnija greška nego nota koja je izostavljena i nije predviđena. Zato kreiranje sveobuhvatnih evaluacijskih metrika i dalje ostaje otvoren problem na području automatske transkripcije glazbe.

6.5. Transkripcija udaračkih instrumenata

Otvoreni problem u području procesiranja glazbenog signala je određivanje i klasifikacija zvukova koji nemaju visinu tona unutar glazbe [9, str. 9]. Najčešće takve zvukove proizvode udarački instrumenti, koji ustvari proizvode široki spektar tonova te s time otežavaju samu detekciju. Tipičan set bubnjeva se sastoji od bas bubnja, doboš bubnja, hi-hat bubnja, tom bubnjeva i činela. Ovakav set bubnjava predstavlja veliki izazov u procesu automatske transkripcije glazbe jer

proizvodi razne harmonične i neharmonične tonove koji često nisu precizno konstantni. Trenutačna rješenja na području AMT-a za udaračke instrumente značajno su bazirana na kreiranju sustava namijenjenog isključivo za transkripciju udaraljki, koji se zatim ukomponiraju u određene postojeće sustave za generalnu transkripciju. Sustavi koji istovremeno transkriptiraju i zvukove koji imaju preciznu visinu tona i zvukove koji nemaju preciznu visinu tona smatraju se robusnim sustavima te postoji vrlo malo, većinom neuspješnih, njihovih pokušaja kreiranja i implementacije.

6.6. Podaci za učenje

Kao i u skoro svim područjima gdje je do sada primijenjeno strojno učenje odnosno umjetna inteligencija tako i u području *AMT*-a samo istraživanje područja podosta pati od nedostatka podataka za učenje. Vrlo je malo dostupnih pjesama kojima je točno odrađena transkripcija od strane ljudskih vještina, te je takva „ručna“ transkripcija podosta vremenski skupa i zahtjeva stručne glazbenike, a treniranje modela na imalo pogrešnim transkripcijama bi uveliko pogoršalo i otežalo sveukupni proces, jer bi modeli počeli zaključivati nepravilnosti kao pravilan način transkripcije [25, str. 20]. Još jedan problem koji valja spomenuti je taj da je puno teže augmentirati (uvećati broj podataka za učenje primjenom jednostavnih transformacija nad podacima) podatke kao što se to može na primjer u strojnom učenju klasifikacija objekata na slikama. Na slikama je moguće provoditi razne transformacije kao što su rotacije, uveličavanje, smanjivanje, filteri s raznim efektima i slično dok se sa zvukom može eventualno automatski promijeniti visina tona pojedinog audio zapisa korištenjem filtara signala. Ubrzavanje ili usporavanje zvučnog zapisa poprilično je komplicirano zbog automatske spontane promjene visine tona prilikom takvih procesa te rješenje takvih problema pripada u druge domene procesiranja glazbenog signala.

7. Zaključak

U ovom radu teorijski su objašnjene značajke glazbe koje su potrebne za uspješan proces automatske transkripcije glazbe. Opisan je odnos frekvencije i visine tona pojedinih nota te je objašnjeno kojim se mjerama određuju dinamika i intenzitet i kako utječu na dojam glazbe. Također opisana je boja zvuka odnosno tona i prikazano je kako se ton vremenski razvija. U opisima razina transkripcije dana su objašnjenja i primjeri pojedinih postupaka automatske transkripcije.

Korištenjem nekih od metoda za obradu signala kao što su konstantna Q transformacija i *Short-time* Fourierova transformacija prikazano je kako se izvlače značajke poput visine tona, početaka i trajanja nota na jednostavnoj monofonijskog glazbi. Iz prikaza usporedbe dvije navedene metode određene su prednosti i nedostatci istih te su opisane situacije u kojima je bolje koristiti koju od metoda. Iako su takve metode prisutne i u naprednim načinima transkripcije, ipak je podosta neprecizno s njima određivati glazbene značajke polifonijske glazbe jer se točnost gubi već pri prisustvu šuma u signalu monofonijske skladbe. Nadalje, objašnjene su više glazbene strukture glazbe poput polifonije i harmonije te njihov usporavajući utjecaj na razvitak potpunih sustava za automatsku transkripciju. Navedene strukture uvećavaju broj strukturnih veza unutar glazbe te time za samu obradu glazbe zahtijevaju kompliciranije sustave koji se većinom baziraju na korištenju neuronskih mreža.

U svrhu praktičnog dijela ovog rada dan je jednostavan algoritam za detekciju akorda unutar skladbe. Algoritam je podijeljen u 8 koraka koji su pojedinačno opisani. Prema danim primjerima izlaza može se zaključiti kako je za preciznu detekciju akorda potreban puno robusniji algoritam jer točnost naglo opada uvođenjem imalo smetnji u zvučni zapis kao što je to na primjer šum. U svrhu poboljšanja algoritma trebalo bi omogućiti raspoznavanje većeg broja akorda te bolje filtriranje ulaznog signala. Gotovo sva današnja rješenja za ovakav problem nisu implementirana samo pomoću standardnih metoda obrade signala, kao što je to ovaj algoritam, već za svoj rad koriste metode strojnog učenja.

Više razine značajki, kao što su to na primjer počeci nota, ritam i tempo, objašnjene su i jednostavnim metodama određene. Ovakve značajke gotovo je nemoguće promatrati kao izolirani slučaj jer su međusobno jako povezane te se osim osnovnih metoda obrade signala moraju koristiti i napredniji načini kako bi se kvalitetno odredile iste. Takvi načini se kod određivanja početaka nota svode na vrlo detaljnu analizu signala, dok se u svrhu određivanja ritma i tempa najčešće

oslanja na uporabu *PLP* funkcija. Osim navedenih načina uvelike se zbog međusobnih strukturalnih veza ovakvih značajki rabe i neuronske mreže.

Napredni i razvijeni sustavi za automatsku transkripciju ili koriste kompliciranе metode kao što je to množenje nenegativnih matrica ili se u potpunosti baziraju na neuronskim mrežama. Kao primjer navedeni su izlazi postupka množenja nenegativnih matrica u kojima se vide bitne značajke za daljnju obradu kao što su to počeci nota i same visine tonova. Na primjeru dvaju modela, koji predstavljaju do sada jedna od najkvalitetnijih rješenja automatske transkripcije, prikazan je trenutačni nedostatak odnosno uspjeh neuronskih mreža pri kompletnom određivanju nota iz glazbenog zapisa. Problem potpune automatske transkripcije nije i još dugo neće biti kompletno usavršen no trenutno postoje rješenja koja daju dovoljno zadovoljavajuće rezultate.

Budućih ciljeva u području *AMT*-a ima puno te je velika većina njih povezana s ostalim područjima računalne obrade signala. Najviše se ističu problemi razdvajanja izvora signala i problemi automatskog prepoznavanja govora koji bi uvelike koristili razvijanju područja automatske transkripcije, a ujedino bi od njega imali i koristi. Od ostalih ciljeva ističe se stvaranje sustava za određivanje glazbenih žanrova. Kao i u skoro svakom području u kojem se koriste neuronske mreže tako i područje *AMT*-a pati od nedostatka podataka za razvijanje modela kao i od načina vrednovanja dobivenih rezultata.

LITERATURA

[1] D. Gerhard, „Computer music analysis“, *Simon Fraser Univ Sch. Comput Sci Surrey UK Tech Rep CMPT TR*, str. 97–13, 1997.

[2] K. D. Martin, „A blackboard system for automatic transcription of simple polyphonic music“, Massachusetts Institute of Technology Media Laboratory Perceptual Computing Section, Cambridge, Technical report 385, 1996.

[3] A. Klapuri, A. Eronen, i T. Virtanen, „Automatic Transcription of Music“, stu. 2001.

[4] J. Cameron, „MIREX“, *MIREX*, 2010. https://www.music-ir.org/mirex/wiki/MIREX_HOME (pristupljeno 25. srpanj 2023.).

[5] M. Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-21945-5.

[6] „Octave“, *Wikipedia*. 09. srpanj 2023. Pristupljeno: 08. kolovoz 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Octave&oldid=1164554848

[7] S. Tjoa, „musicInformationretrieval.com audio_representation“. https://colab.research.google.com/github/stevetjoa/musicinformationretrieval.com/blob/gh-pages/audio_representation.ipynb#scrollTo=8cVhnL8H6YZC (pristupljeno 08. kolovoz 2023.).

[8] „MusicXML“, *Wikipedia*. 25. srpanj 2023. Pristupljeno: 26. srpanj 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=MusicXML&oldid=1167024638

[9] E. Benetos, S. Dixon, Z. Duan, i S. Ewert, „Automatic Music Transcription: An Overview“, *IEEE Signal Process. Mag.*, sv. 36, izd. 1, str. 20–30, sij. 2019, doi: 10.1109/MSP.2018.2869928.

[10] „Spectrogram - Wikipedia“. https://en.wikipedia.org/wiki/Spectrogram (pristupljeno 10. kolovoz 2023.).

[11] „librosa — librosa 0.10.1 documentation“. https://librosa.org/doc/latest/index.html (pristupljeno 28. kolovoz 2023.).

[12] S. Tjoa, „musicInformationretrieval.com fourier_transform“. https://colab.research.google.com/github/stevetjoa/musicinformationretrieval.com/blob/gh-pages/fourier_transform.ipynb#scrollTo=QrFlat0JEx2o (pristupljeno 10. kolovoz 2023.).

[13] „SciPy“. https://scipy.org/ (pristupljeno 28. kolovoz 2023.).

[14] „Short-time Fourier transform“, *Wikipedia*. 11. listopad 2022. Pristupljeno: 10. kolovoz 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Short-time_Fourier_transform&oldid=1115436538

[15] „Constant-Q transform“, *Wikipedia*. 22. srpanj 2023. Pristupljeno: 10. kolovoz 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Constant-Q_transform&oldid=1166609682

[16] P. E. Matthaei, „Automatic music transcription: An exploratory study“, Stellenbosch University, Stellenbosch, 2004.

[17] „Chroma feature“, *Wikipedia*. 27. srpanj 2023. Pristupljeno: 11. kolovoz 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Chroma_feature&oldid=1167356737

[18] „AI-complete“, *Wikipedia*. 06. kolovoz 2023. Pristupljeno: 13. kolovoz 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=AI-complete&oldid=1168939369

[19] „Rhythm“, *Wikipedia*. 19. svibanj 2023. Pristupljeno: 29. kolovoz 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Rhythm&oldid=1155728303

[20] „Tempo“, *Wikipedia*. 27. kolovoz 2023. Pristupljeno: 29. kolovoz 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Tempo&oldid=1172558187

[21] „Artificial intelligence“, *Wikipedia*. 30. kolovoz 2023. Pristupljeno: 31. kolovoz 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Artificial_intelligence&oldid=1173002442
[22] „Artificial neural network“, *Wikipedia*. 17. kolovoz 2023. Pristupljeno: 31. kolovoz 2023. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=1170800588
[23] „Music Transcription with Transformers“. https://magenta.tensorflow.org/transcription-with-transformers (pristupljeno 31. kolovoz 2023.).
[24] „AnthemScore - Automatic Music Transcription Software“. https://www.lunaverus.com/ (pristupljeno 01. rujan 2023.).
[25] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, i A. Klapuri, „Automatic music transcription: challenges and future directions“, *J. Intell. Inf. Syst.*, sv. 41, izd. 3, str. 407–434, pros. 2013, doi: 10.1007/s10844-013-0258-3.

Sažetak

Naslov: Automatska transkripcija glazbe

Sažetak: Problem automatske transkripcije glazbe poznat je znanstvenicima i istraživačima još od 70-ih godina prošlog stoljeća. Iako se čini kao trivijalan zadatak kojemu je cilj pretvorba zvučnog zapisa u notni zapis, zapravo predstavlja široko područje primjene. Područje problema automatske transkripcije podosta je zanimljivo jer objedinjuje glazbene i računalne znanosti te otvara nove mogućnosti proširivanja područja istraživanja obrade zvučnih signala. U ovom radu opisane su glavne značajke glazbe potrebne za proces automatske transkripcije. Opisane su razine automatske transkripcije kao i više glazbene strukture poput harmonije i polifonije. Dani su programski primjeri određivanja visina tonova, ritma, tempa i početaka nota zajedno s vizualnim prikazima izlaza takvih primjera. U svrhu praktičnog dijela stvoren je algoritam za detekciju akorda unutar skladbe. Na primjeru algoritma se jasno primjećuje kako je za usavršavanje detektiranja glazbenih struktura potrebno imati robustan algoritam iz kojeg stoji puno istraživanja. Također su prikazane i uspoređene napredne metode automatske transkripcije koje se koriste postupcima kao što su to množenje nenegativnih matrica i uporabom neuronskih mreža.

Ključne riječi: analiza signala, automatska transkripcija glazbe, detekcija akorda, neuronske mreže, određivanje nota.

Abstract

Title: Automatic music transcription

Abstract: Automatic transcription problem is knows to scientists and researchers since the 70s of the last century. Although it seems like a trivial task which goal is to convert audio record into a music notation it actually represents wide area of use. The scope of automatic music transcription problem is quite interesting because it unites musical and computer branches of science while opening new possibilities of expanding research area in terms of processing audio signals. This paper describes main features of music needed for the process of automatic transcription. Levels of automatic transcription are described as well as higher musical structures such as harmony and polyphony. Programming examples of determining pitches, rhythm, tempo, and note onsets are provided along with visual representations of the outputs of such examples. For practical purposes of this paper, an algorithm was created for the detection of chords within a composition. The example of the algorithm clearly shows that in order to perfect the detection of musical structures, it is necessary to have a robust algorithm, which is based on a lot of research. Also presented and compared are advanced methods of automatic transcription that use procedures such as non-negative matrix multiplication and the use of neural networks.

Keywords: automatic music transcription, chord detection, neural networks, note detection, signal analysis.