UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

MASTER RAD

STEFAN KOSTIĆ

TRANSKRIPCIJA MONOFONIH MELODIJA

Beograd, 2017.

Mentor: dr Saša Malkov, *Matematički fakultet*

Članovi komisije:

dr Filip Marić, *Matematički fakultet*

dr Nenad Mitić, *Matematički fakultet*

Datum odbrane: _____

Sažetak

Kada muzičar sluša neki muzički komad i zapisuje note koje ga čine, u muzičkoj notaciji, onda se kaže da je on napravio transkripciju tog muzičkog dela. U tradicionalnom smislu, obično se kao rezultat transkripcije dobija partitura. Proces transkribovanja zahteva vreme i muzičko obrazovanje, kao i dosta iskustva. Automatska transkripcija zvuka može biti jako bitna olakšica muzičarima koji uvežbavaju svoju sposobnost transkribovanja na osnovu sluha, ali i iskusnijim muzičarima kao provera ili ubrzavanje procesa. U ovom radu opisani su neki algoritmi i metode za automatsku transkripciju monofonih melodija. Kao rezultat rada, implementirana je aplikacija koja kombinuje opisane tehnike i dati su rezultati rada implementiranog sistema.

Ključne reči: automatska transkripcija, monofone melodije, detektovanje visine tona, detektovanje trajanja tona, radni okvir .NET, biblioteka NAudio

Sadržaj

Poglavlje 1

Uvod

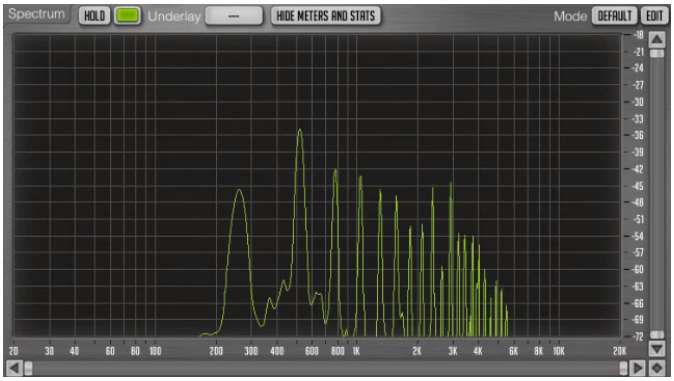
1.1 Zvuk

Zvuk predstavlja zvučni talas koji stvara neki izvor zvuka. Izvor zvuka može biti gotovo bilo šta, od muzičkog instrumenta i zvučnika do govora ljudi, laveža pasa i slično. Ukoliko se nalazi u nekoj sredini koja prenosi mehaničke tj. zvučne talase, kao što su voda ili vazduh, onda izvor stvara vibracije koje se prenose dalje od izvora brzinom prenošenja zvuka u toj sredini, stvarajući na taj način zvučne talase. Ljudi imaju sposobnost percipiranja i prepoznavanja tih talasa pomoću složenih mehanizama u ušima i mozgu. Kada zvučni talasi prođu kroz ušni kanal, prouzrokuju vibraciju bubne opne. Ta vibracija kasnije dovodi do pomeranja tečnosti u unutrašnjem uhu i zatim do prevođenja vibracija u nervne impulse. Nervni impulsi se preko nerava prenose do mozga gde se prepoznaju kao ono što čujemo.

Postoje četiri najbitnije fizičke karakteristike zvuka [16]: učestalost (ili frekvencija), tj. broj ponavljanja vibracija u sekundi, amplituda vibracije, oblik vibracije i položaj izvora zvuka u odnosu na slušaoca. Njima odgovoraju subjektivne karakteristike: visina zvuka, jačina zvuka, boja zvuka i prostorna percepcija. Zvuk koji ima prepoznatljivu i jasnu visinu, jačinu, boju i oblik naziva se ton. Prva karakteristika, učestalost ili broj ponavljanja vibracija u sekundi (eng. *vibration repetition rate*) naziva se još i fundamentalna frekvencija i označava sa F_0. Za periodične signale F_0 se definiše kao inverzna vrednost trajanja perioda (izraženog u sekundama) koja može biti definisana kao najmanji pozitivni član beskonačnog skupa vremenskih pomeranja koje ostavljaju signal u nepromenljivom stanju.

Ako je učestalost ponavljanja vibracija u sekundi veća, ljudi percipiraju taj zvuk ili ton višim, i obrnuto, što je učestalost manja, ton je percipiran kao niži. Dakle, može se reći da frekvenciji odgovara subjektivna karakteristika tona koja se naziva visina tona (eng. *pitch*). Iako se fundamentalna frekvencija definiše samo za periodične signale, visina tona i fundamentalna frekvencija su često u literaturi sinonimi, pa će tako biti i u ovom radu. Ono što je jako bitno napomenuti jeste da se mnogi tonovi koji su složeniji po svojoj prirodi sastoje iz harmonika (eng. *harmonics*). Harmonici su neki umnožak fundamentalne frekvencije. Na slici 1.1 korišćenjem alata firme Voxengo [24] može se videti da je ton C odsviran na

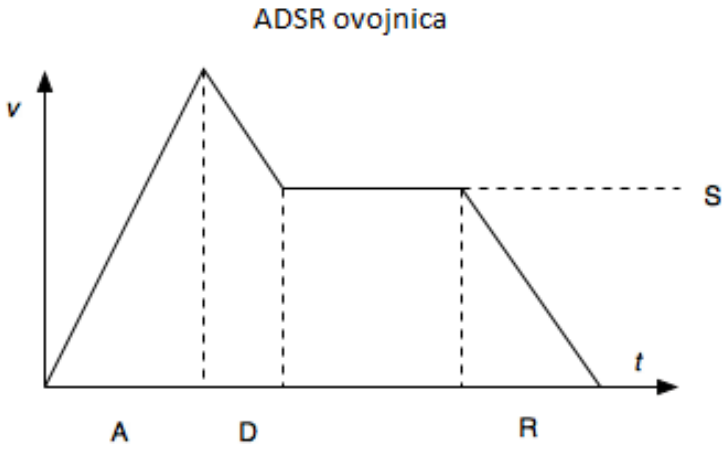
klaviru jako bogat harmonicima. Takođe, vidi se da je jedna od istaknutijih frekvencija negde na 261Hz (što odgovara fundamentalnoj frekvenciji srednjeg C).



Slika 1.1: Spektar tona C odsviranog na klaviru

Sledeća bitna fizička karakteristika zvuka jeste amplituda vibracije i njoj odgovara subjektivna karakteristika koja se naziva jačina zvuka.

Obliku vibracija odgovara boja zvuka (eng. *timbre*). Zahvaljujući boji zvuka, ljudsko uvo može da razlikuje tonove koji imaju istu visinu i jačinu. Postoje mnoga svojstva koja određuju boju, a neka od najbitnijih su harmonici, tremolo, vibrato, ali i elementi koji čine ovojnicu (eng. *envelope*) zvuka: napad, opadanje, zadržavanje i otpuštanje, tzv. ADSR (eng. *attack-decay-sustain-release*) model o kome će biti malo više reči u nastavku rada. ADSR model je dat na slici 1.2.



Slika 1.2: ADSR ovojnica

Poslednja bitna fizička karakteristika zvuka jeste položaj izvora zvuka u odnosu na primaoca zvuka što odgovara prostornoj percepciji. Prostorna percepcija se obično omogućava

obezbeđivanjem višekanalnog zvuka, pri čemu se kasnije svaki kanal emituje iz posebnog izvora.

1.2 Muzička notacija zvukova i melodija

Imena tonova ili nota su uvedena zbog lakše muzičke komunikacije između ljudi. Ime tona ukazuje na njegovu visinu, pa tako imamo, po abecedi:

C, D, E, F, G, A, H

ili po solmizaciji:

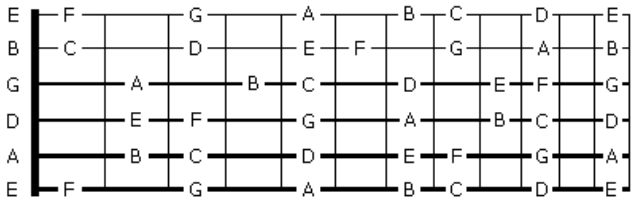
Do, Re, Mi, Fa, Sol, La, Si

U prvom slučaju su u pitanju apsolutne oznake, dok je osnovna frekvencija tona A 440Hz. U drugom slučaju je reč o relativnim oznakama. Note se obično zapisuju od nižih ka višim u notnom zapisu tako da je određivanje njihove visine vizuelno i prilično lako. Ljudi i instrumenti obično mogu da proizvedu više od sedam tonova, zbog čega su tonovi grupisani u oktave. Skup svih tonova zove se tonski sistem.

$_2$C $_2$D $_2$E $_2$F $_2$G $_2$A $_2$H	Subkontra oktava
$_1$C $_1$D $_1$E $_1$F $_1$G $_1$A $_1$H	Kontra oktava
C D E F G A H	Velika oktava
c d e f g a h	Mala oktava
c^1 d^1 e^1 f^1 g^1 a^1 h^1	Prva oktava
c^2 d^2 e^2 f^2 g^2 a^2 h^2	Druga oktava
c^3 d^3 e^3 f^3 g^3 a^3 h^3	Treća oktava
c^4 d^4 e^4 f^4 g^4 a^4 h^4	Četvrta oktava
c^5 d^5 e^5 f^5 g^5 a^5 h^5	Peta oktava

Tabela 1.1: Oktave tonskog sistema

Niz tonova čini skalu ili lestvicu. Tako C, D, E, F, G, H, C čine osnovnu lestvicu, C dur lestvicu. Svaki ton je jedan od stupnjeva skale. Između stupnjeva lestvice su razmaci koji se zovu intervali ili stepeni. Tako kod C dur lestvice između C i D je ceo stepen, a između E i F polustepen.



Slika 1.3: C dur skala na gitari

Interval između dva susedna tona je polustepen. Ton može biti povišen za polustepen, i to se zapisuje pomoću povisilice (najčešće simbol ♯) ili može biti snižen za polustepen,

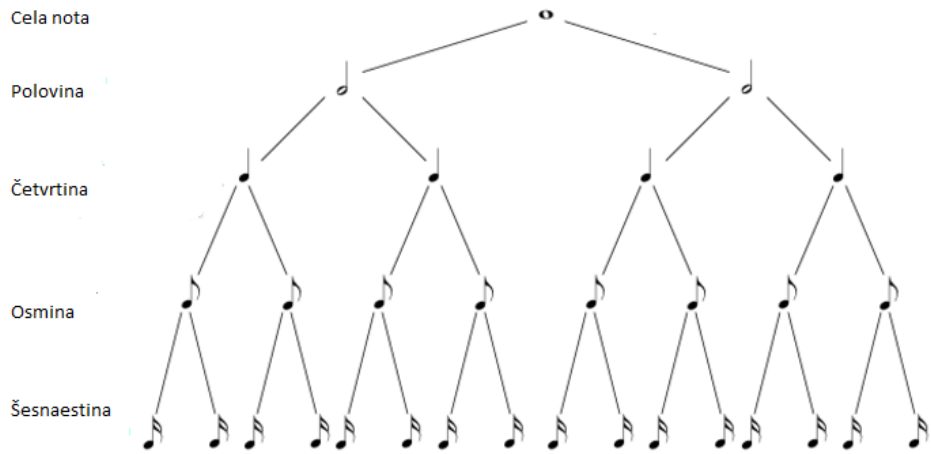
što se zapisuje pomoću snizilice (najčešće simbol ♭). Tako povišen ton u imenu dobija dodatak -is (eng. *sharp*), a sniženi -es (eng. *flat*). Tako se povišeno C čita Cis, a sniženo Ces. Pošto je između E i F polustepen, obično se ne koristi Eis ili Fes. Isto tako, u centralnoj i istočnoj Evropi, kao i u Skandinaviji, se sniženo H naziva B. U Americi, Kanadi, Australiji, Britaniji, Irskoj i Holandiji sa B se označava poluton ispod C, a sa B♭ (eng. *B flat*) ceo ton ispod C.

Percepcija tonova je vezana za eksponencijalnu promenu frekvencija. Frekvencije 12 tonova koji su navedeni na početku ovog poglavlja mogu se dobiti pomoću formule:

$$f = 2^{\frac{x}{12}} f_b$$

Sa f_b označena je bazna frekvencija, i to je obično frekvencija tona A koja iznosi 440Hz. Međunarodna agencija za standardizaciju (eng. *International Organization for Standardization*) navodi ovu frekvenciju kao standardnu za štimovanje [11]. Sa x je označeno koliko polutonova je udaljen ton za koji se računa frekvencija od baznog tona. Odnos između dve oktave je 1:2.

Grupisani tonovi čine melodije i akorde. U ovom radu posebno su zanimljive monofone melodije u kojima se svira jedan ton u jednom trenutku. Suprotno monofonim su polifone melodije. Akord čini tri ili više tonova odsviranih odjednom.



Slika 1.4: Ritmičko drvo

Svaki ton ima svoje trajanje. Razlikujemo celu notu, polovinu, četvrtinu, osminu, šestanestinu, tridesetdvojku, šestdesetčetvortku. U ovom nizu svaki sledeći traje duplo manje od prethodnog. Ovaj odnos se najbolje vidi na ritmičkom drvetu [10]. Primer ritmičkog drveta dat je na slici 1.4. Postoje i trajanja sa tačkom. Ukoliko se je u pitanju nota sa tačkom onda ona traje duže za pola originalne dužine.

Otkucaj (eng. *beat*) je osnovna jedinica merenja ritma. Broj otkucaja u minuti (eng. *beats per minute*) je brzina muzike, odnosno tempo. Muzika je često podeljena u taktove jednakih dužina. Najpopularniji takt je 4/4 iz čega sledi da je dužina jednog takta jednaka dužini četiri nota četvrtina.

1.3 Digitalni zvuk - osnovni pojmovi

1.3.1 Zvučni signal

Zvučni signal je reprezentacija zvuka i matematički se može predstaviti funkcijom *f(t)* koja označava promenu vazdušnog pritiska usled zvuka u vremenskom trenutku *t*. Ukoliko su domen i kodomen ove funkcije neprekidni, onda je u pitanju analogni signal. Ako su domen i kodomen diskretni skupovi vrednosti onda govorimo o digitalnom signalu.

Digitalni signal ima mnoge prednosti u odnosu na analogni. Digitalni CD je mnogo robusniji od analogne ploče jer na digitalne signale ne utiču nesavršenosti elektronskih sistema koje kvare analogne signale. Isto tako, mnogo su imuniji na šum, mogu da se kodiraju, sigurniji su, koriste manji protok, lakši su za prenos. Takođe, mogućnosti za njihovo procesiranje od strane računara su jako velike.

Dakle, digitalni signal je jako pogodan i koristan, tako da je često potrebna konverzija iz analognog u digitalni signal. Za konverziju se koriste specijalni uređaji koji se zovu ADC (eng. *analog-to-digital*) konverteri. Uređaji koji vrše obrnutu konverziju zovu se DAC (eng. *digital-to-analog*) konverteri. ADC i DAC konverteri se nalaze i u zvučnim kartama.

1.3.2 Digitalizacija

ADC konverter je zadužen za digitalizaciju domena i kodomena analognog signala.

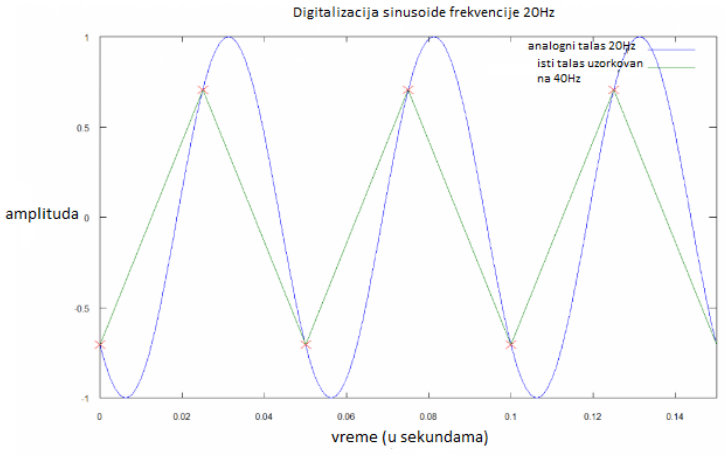
Uzorkovanje

Digitalizacija domena naziva se uzorkovanje ili semplovanje (*eng. sampling*). Uzorkovanjem digitalizujemo informacije dobijene iz analognog signala. Proces uzorkovanja je prilično jednostavan i zasniva se na tome da se mere vrednosti signala u pravilnim vremenskim intervalima. Glavno pitanje i problem kod uzorkovanja jeste učestalost uzorkovanja (eng. *sample rate*), odnosno, koliki treba da bude vremenski interval između dva uzorka da bi uzorkovanje bilo pravilno. Ukoliko je moguće rekonstruisanje analognog signala iz uzoraka, onda možemo da kažemo da je uzorkovanje pravilno. Odgovor na ovo pitanje daje *teorema o uzorkovanju* koja se još naziva i Najkvistova ili Šenonova teorema [20].

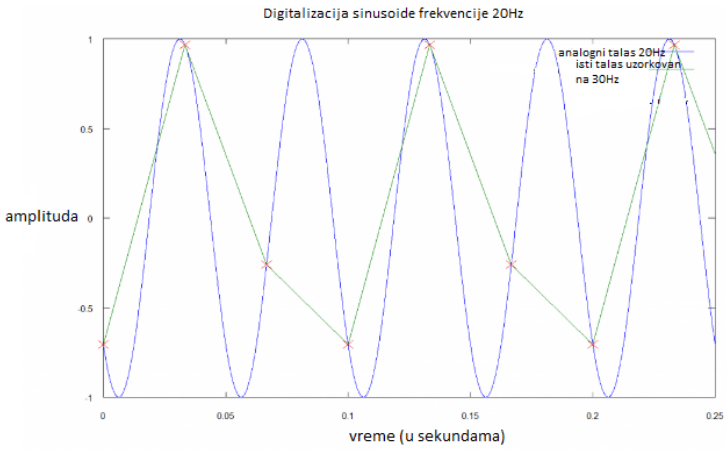
Teorema 1 (Teorema o uzorkovanju)**.** *Ukoliko je f najveća frekvencija neke od komponenti posmatranog analognog signala, onda učestalnost uzorkovanja mora biti bar 2f da bi se originalni signal mogao verno rekonstruisati iz uzoraka.*

Dakle, ukoliko je frekvencija uzorkovanja bar dva puta veća od najveće frekvencije signala, imaćemo bar dve tačke po ciklusu originalnog signala, što je dovoljno za rekonstrukciju [12]. Međutim, u praksi je ipak potrebno da se ide i malo dalje od toga, jer nesrećnim izborom trenutka u kome se počinje uzorkovanje može se dobiti uzorak koji ne omogućava vernu rekonstrukciju početnog analognog signala. Na primer, na slici 1.5 vidi se uzorkovanje dvostrukom učestalošću, koje ipak ne omogućava vernu rekonstrukciju početnog signala jer nije sačuvana amplituda signala.

Primer uzorkovanja signala koji nije dosledan Najkvistovoj teoremi dat je na slici 1.6, gde se vidi da je frekvencija uzorkovanog signala manja od frekvencije originalnog. Kada



Slika 1.5: Uzorkovanje zvuka frekvencije 20Hz učestalošću uzorkovanja od 40Hz



Slika 1.6: Uzorkovanje zvuka frekvencije 20Hz učestalošću uzorkovanja od 30Hz

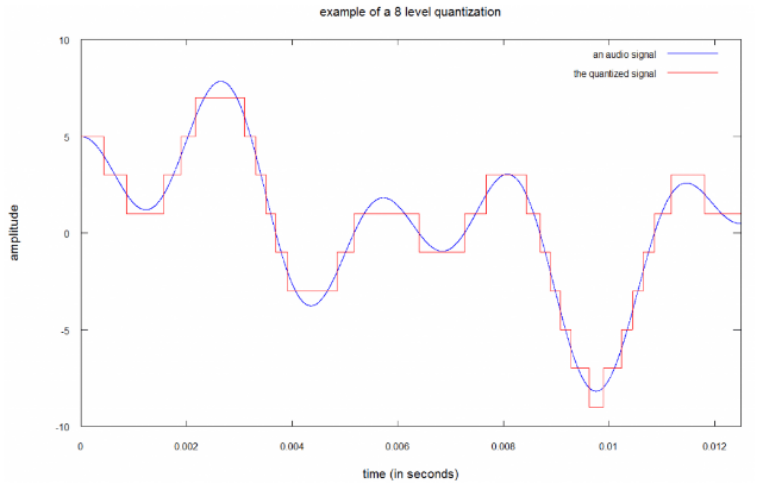
se visoke frekvencije uzorkuju previše malom učestalošću uzorkovanja dolazi do pojave koja se naziva alijasing (eng. *aliasing*). Ovaj problem može da se reši npr. korišćenjem niskopropusnog filtera (eng. *low pass filter*) koji ne propušta ili seče frekvencije koje su iznad najviše frekvencije. To često automatski rade zvučne karte, kao i mnogi dostupni alati.

Najpopularnija učestalost uzorkovanja koja se koristi u praksi je 44,1KHz (CD zvuk). Pored toga, koristi se i 48KHz (DVD), ali i mnogo veće učestalosti koje se koriste u studijskim produkcijama, kao što su 96KHz ili čak 192KHz. Niže učestalosti od 8KHz i 16KHz koriste se npr. u telefoniji. Ako se ima u vidu da je prosečan opseg frekvencija koje čovek može da čuje između 20Hz i 20KHz, učestalost od 44,1KHz zadovoljava Najkvistovu teoremu i u mnogim slučajevima predstavlja optimalnu učestalost. Prevelike

učestalosti dovode do više zauzetog prostora, što sadržaje koji su na ovaj način uzorkovani čini, između ostalog, i nepogodnim za prenošenje preko Interneta, a naročito za prenos uživo (eng. *live streaming*). Takođe, postoje tvrdnje da učestalost preko 96KHz može da dovede do neželjene iskrivljenosti (eng. *distorsion*) zvuka [6].

Kvantizacija

Digitalizacija jačine zvuka signala naziva se kvantizacija. Glavno pitanje u ovom domenu je određivanje nivoa kvantizacije, odnosno odabir broja bitova koji će se koristiti za čuvanje jednog uzorka. Ukoliko koristimo 8-bitne uzorke, dobijamo lošiji kvalitet zbog niskog odnosa između signala i šuma (eng. *signal to noise ration*) i mogućnosti da se čuva samo 256 različitih vrednosti. Pošto je cilj da signal bude što glasniji a šum neprimetniji, u praksi se koriste 16-bitni i 24-bitni uzorci. Pored dobrog odnosa signala i šuma i većeg opsega vrednosti, na ovaj način dobija se i mnogo veći dinamički opseg koji predstavlja razliku između najglasnijeg i najtišeg zvuka koji možemo da predstavimo [12].

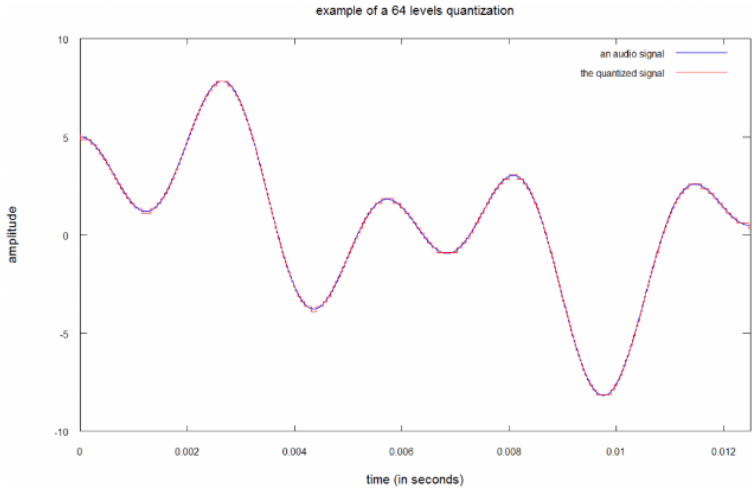


Slika 1.7: Kvantizacija sa 3 bita (8 nivoa) po uzorku

Dinamički opseg se meri u decibelima. Decibeli su relativna mera i označavaju se u odnosu na odabrani referentni nivo. Merna jedinica koja se koristi je dB. Tako npr. 0dB označava da zvuk ima isti nivo kao referentni, +3dB da je malo glasniji, -3dB da je malo tiši. Kod digitalnog zvuka, obično se 0dB dodeli najvećem mogućem nivou, pa je to kod 8-bitnog uzorka 255, a kod 16-bitnog 32767. Ako sa P označimo vrednost amplitude uzorka, a sa P_0 referentnu vrednost, decibeli se računaju po formuli:

$$10log_{10}(P/P_0) \tag{1.1}$$

Kao što se vidi, u pitanju je logaritamska skala. Ukoliko je vrednost uzorka duplo veća od referentne dobijamo +3dB, a ukoliko je duplo manja onda -3dB. Ekvivalentno, ukoliko se broj bitova koji se koristi za zapis uzorka poveća za 1, dobija se duplo više vrednosti



Slika 1.8: Kvantizacija sa 6 bitova po uzorku

koje mogu da se zapišu ili dinamički opseg veći za 3dB. Tako, ako se koristi 16 bitova dobija se dinamički opseg od 48dB.

Povećavanjem broja bitova smanjuje se i greška kvantizacije (razlika između stvarnog signala i onog koji se dobija kvantizacijom). Za smanjivanje ove greške pored povećavanja broja bitova, koristi se i dodavanje belog šuma (eng. *dithering*) signalu [20]. Beli šum je buka koja se dobija kombinovanjem zvukova različitih frekvencija. Ukoliko se iskombinuju sve frekvencije koje čovek može da čuje dobija se beli šum.

U profesionalnoj muzičkoj produkciji uglavnom se koristi 24 bita za čuvanje uzoraka, što daje dinamički opseg od 144dB. Međutim, pravi razlog za upotrebu 24 bita najčešće leži u tome što se na taj način izbegava sečenje signala iznad maksimalne vrednosti (eng. *clipping*) jer se na ovaj način dobije mnogo dodatnog prostora (eng. *headroom*). Ako se radi sa 16 bita onda se pri svakom snimanju zvuka mora voditi računa o tome da se maksimalno iskoristi raspoloživ dinamički opseg. Sa druge strane, kada se koristi 24 bita dovoljno da se pri snimanju upotrebi vrlo mali deo dinamičkog raspona a da zapis i dalje bude dovoljno detaljan. Takođe, omogućavanjem više nivoa se smanjuje uticaj računskih grešaka na preciznost zvučnog zapisa i vernost reprodukcije.

U progamiranju zvučnih aplikacija se najčešće koristi 32 bita za zapis jednog uzorka zbog karakteristika programskih jezika koji nemaju tip podataka od 24 bita. Takođe, veoma često se koriste decimalni brojevi normalizovani između -1 i +1, i tip koji se u programskim jezicima uglavnom naziva *float*. Glavni razlozi su udobnost operacija prilikom digitalne obrade zvuka, ali i izbegavanje sečenja signala. Ako se npr. kao referentna vrednost uzme +1, onda zvuk može da ide i preko 0dB.

1.3.3 Kanali

Kanali predstavljaju broj ulaznih signala koji se uzorkuju. Kada je u pitanju muzika, najčešće su to mono (1 kanal) ili stereo (2 kanala). Kod mono sistema, svi zvučni signali

su pomešani (eng. *mixed*) kroz jedan kanal. Ukoliko je mono sistem dobro dizajniran, svaki slušalac čuje zvuk suštinski na istom nivou što mono sisteme čini jako dobrim za razumljivost govora. Stereo sistemi imaju dva nezavisna zvučna kanala i koriste se kada je potrebno replicirati slušnu perspektivu i lokalizaciju instrumenata i vokala. Stereo često predstavlja najbolji izbor za slušanje muzike. Postoje i okružujući sistemi (eng. *surrounding system*) koji koriste veći broj kanala i jako su pogodni za zvuk tokom gledanja filmova. Ograničavajući faktor kod broja ulaznih i izlaznih kanala je pretežno zvučna karta.

1.3.4 Formati zvučnih datoteka

Postoje različiti formati audio datoteka. Razlikuju se komprimovani i nekomprimovani formati, i u zavisnosti od toga se razlikuju i informacije koje se skladište u njima. Tako npr. i komprimovani i nekomprimovani sadrže deo o informacijama zvučnog formata, ali su kod nekomprimovanih to često informacije kao što su učestalost uzorkovanja, broj bitova po uzorku, broj kanala, dok su kod komprimovanih to informacije o kodiranju, odnosno o kodeku (eng. *codec*) i njegovi parametri kao što je broj bitova u sekundi (eng. *bitrate*). Mogu da sadrže i opcione metapodatke (eng. *metadata*) koji sadrže informacije o imenu autora, imenu pesme, naslov albuma i slično.

Nekomprimovani formati

Kao što samo ime kaže, ovi formati sadrže uzorke koji nisu komprimovani ni na koji način. Metoda kodiranja se zove pulsna kod modulacija (PCM). Prirodno, ovi formati zauzimaju dosta mesta na disku i nisu pogodni za prenos. Takođe, pošto su informacije o formatu na početku, obrada ovakvih datoteka nije moguća od proizvoljne pozicije. Najpoznatiji je format WAVE ili WAV (*Waveform Audio File Format*). Pored formata WAV postoje i mnogi drugi nekomprimovani formati kao što je Eplova alternativa format AIFF (*Audio Interchange File Format*). Pošto je maksimalna veličina WAV datoteke 4GB postoji i format RF64 koji je duplo veći.

Format WAV

Format WAV je standardni format audio datoteka koji je originalno napravljen od strane Majkrosofta (eng. *Microsoft*) i IBM-a 1991. godine za operativni sistem Vindouz (eng. *Windows*), ali se vremenom ustalio kao najpopularniji i na ostalim operativnim sistemima.

Format WAV je instanca generičkog formata RIFF (*Resource Interchange File Format*) u kome se podaci čuvaju u vidu parčadi (eng. *chunks*).

WAV datoteka sadrži RIFF zaglavlje, parče koje sadrži informacije o formatu, parče za podatke, i opciono, parčad za metapodatke.

RIFF zaglavlje se sastoji od 12 bajtova [14] i to:

Sadržaj	Veličina u bajtovima
"RIFF"	4
Veličina parčeta	4
"WAVE"	4

Tabela 1.2: RIFF zaglavlje

Parče sa podacima izgleda ovako [14]:

Sadržaj	Veličina u bajtovima
"data"	4
Veličina parčeta	4
Podaci	

Tabela 1.3: Parče sa podacima

Parče koje sadrži informacije o formatu se sastoji od 26 bajtova [14] i to:

Sadržaj	Veličina u bajtovima
"fmt "	4
Veličina format parčeta	4
Audio format	2
Broj kanala	2
Učestalost uzorkovanja	4
Broj bajtova po sekundi	4
Poravnanje po bloku	2
Broj bitova po uzorku	2
Opcionalni dodatni bajtovi	2

Tabela 1.4: Parče koje sadrži informacije o formatu

Komprimovani formati

Komprimovani formati nastaju korišćenjem dela kodeka koji se naziva koder. Cilj komprimovanja jeste da se uz što manji procenat gubljenja na kvalitetu smanji veličina datoteka. Ovako dobijene datoteke su idealne za prenos, naročito za prenos uživo putem Interneta. Komprimovanje može da bude sa gubljenjem informacija (eng. *lossy*) ili bez gubljenja informacija (eng. *lossless*). Kodeci koji se koriste za komprimovanje sa gubljenjem informacija odbacuju podatke na pametan način, odnosno na način na koji uši i mozak percipiraju zvuk koji ljudi čuju. Istraživanje [17] u kome su ispitanici upoređivali koder AAC koji koristi iTunes i CD kvalitet pokazuje da je ljudima praktično nemoguće da primete razliku. Izuzeci su se dešavali u slučaju da se razlike namerno traže vrlo pažljivim preslušavanjem ili korišćenjem odlične opreme. Takođe, ispitanik koji je imao najbolje rezultate je tražio karakteristične elemente i instrumente u primerima.

Najpoznatiji formati bez gubljenja informacija su FLAC (eng. *Free Lossless Audio Codec*), ALAC (eng. *Apple Lossless Audio Codec*) i slični. Najpoznatiji formati sa gubljenjem informacija su MP3 (eng. *MPEG-1 or MPEG-2 Audio Layer III*), AAC (*Advanced Audio Coding*) i slični.

Format MP3

Format MP3 format je najpopularniji komprimovani format sa gubljenjem informacija. U njemu su sadržani metapodaci (ID3 ili ID3v2 oznaku) i okviri (eng. *frame*). Svaki okvir sadrži i podatke i informacije o formatu, tako da mogu da se dekodiraju počev od bilo koje pozicije. Broj bitova u sekundi varira, ali su najpopularniji 128Kbps, 192Kbps i 320Kbps.

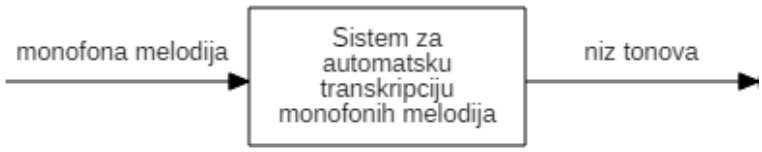
Format FLAC

Format FLAC je najpopularniji komprimovani format bez gubljenja informacija. Za njegovo dobijanje se koristi algoritam FLAC za komprimovanje koji može da smanji veličinu datoteke i preko polovine. Dekomprimovanje dovodi do identične kopije originalnog zvuka. Slično kao i format MP3, sadrži okvire i jako fleksibilne mogućnosti za metapodatke.

1.4 Značaj automatske transkripcije

Automatska transkripcija ima mnogo primena. Pored toga što ubrzava i olakšava proces transkripcije muzičarima, može biti i odlična opcija za učenje i uvežbavanje ove veštine kroz proveru dobijenih rezultata. Automatska transkripcija isto tako pruža dosta informacija o zvuku koji se analizira (visine tonova, njihovo trajanje) i samim tim može biti alat za različite muzičke analize. Jedan od takvih primera dao je MekLeod [5], u čijem radu se navodi da je sistem koji koristi detektovanje visine tonova zahvaljujući vizuelizaciji u realnom vremenu pomogao violinistima u pozicioniranju gudala tokom sviranja. Takođe, automatska transkripcija može biti deo različitih drugih muzičkih aplikacija.

Kada se tonovi nižu jedan za drugim onda čine melodiju. Melodija predstavlja jako bitan deo gotovo svakog muzičkog dela. Ukoliko se u svakom trenutku svira samo jedan ton onda je reč o monofonim melodijama. Na slici 1.9 prikazan je zadatak automatske muzičke transkripcije monofonih melodija. U ovom radu prikazani su različiti algoritmi i tehnike pomoću kojih je moguće razviti sistem koji rešava ovaj zadatak. Takođe je implementiran osnovni sistem za transkripciju koji koristi opisane metode.



Slika 1.9: Zadatak automatske muzičke transkripcije

Poglavlje 2

Automatska muzička transkripcija

2.1 Algoritmi za detekciju visine tona

Visina tona je glavni parametar svake muzičke transkripcije. Zato postoji dosta algoritama i radova na ovu temu. Dve velike grupe algoritama koje se mogu izdvojiti su algortmi u vremenskom domenu, i algoritmi u frekvencijskom domenu. U vremenskom domenu se analizira kako se amplituda signala menja tokom vremena. U frekvencijskom domenu signal se ne analizira u odnosu na vreme već u odnosu na frekvencije i tom analizom može se utvrditi koliko signala se nalazi u kom frekvencijskom opsegu. Prelazak iz vremenskog u frekvencijski domen (i obrnuto) moguć je upotrebom Furijeovih transformacija. Postoje još neke grupe algoritama, npr. algoritmi bazirani na modelima ljudskog uha.

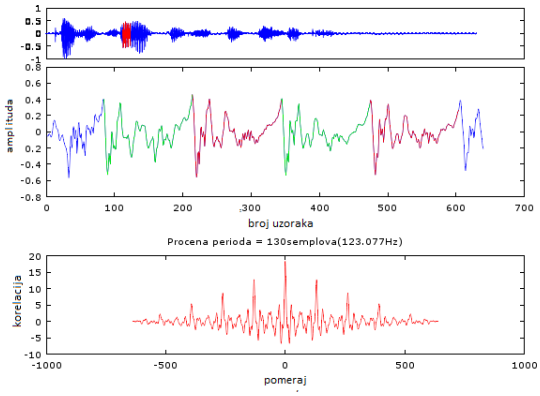
2.1.1 Autokorelacija

Autokorelacija periodičnog signala x_t pokazuje njegovu sličnost sa njegovom kopijom koja kasni u vremenu za neki pomeraj (eng. *lag*) τ. Može se definisati na sledeći način [5] [3]:

$$r_t(\tau) = \sum_{i=t+1}^{t+W} x_i x_{i+\tau}$$

Ovde je W veličina prozora integracije. U kontekstu programiranja i obrade zvučnog signala prozor se naziva bafer (eng. *buffer*) koji je u suštini niz uzoraka. Obrada signala se realizuje bafer po bafer. Neretko se susedni baferi preklapaju kako bi poboljšali preciznost rezultata. U nekim algoritmima se koriste preklapanja od 50% ili 75%. U procesiranju signala koristi se i druga definicija autokorelacije [5] [3]:

$$r_t^{'}(\tau) = \sum_{i=t+1}^{t+W-\tau} x_i x_{i+\tau}$$



Slika 2.1: Ilustracija autokorelacije u programu Octave

U drugoj definiciji može se primetiti da se prozor integracije sužava sa povećanjem vrednosti pomeraja τ.

Autokorelacija ima vrhove u mestima gde su množioci perioda, zbog toga što je za te vrednosti pomeraja sličnost delova koji se preklapaju najveća, tako da postoji način da se period utvrdi pretragom, a samim tim i fundamentalna frekvencija. Na slici 2.1 detektovan je period dužine 130 uzoraka što odgovara frekvenciji 123.077Hz (najbliži ton je H koji ima frekvenciju 123.47Hz). Međutim, autokorelacija nije savršen metod. Ako je signal takav da sličnost postoji i na pola perioda autokorelacijom često može da se pronađe manji period od stvarnog. Takođe, autokorelacija je osetljiva na porast amplituda i sa porastom amplituda često i detektovani vrhovi rastu umesto da ostanu konstantni. Autokorelacija u praksi pokazuje dosta grešaka tako da i pored svoje jednostavnosti i efikasnosti često nije pravi izbor.

2.1.2 MekLaudov algoritam (MPM)

Ovaj algoritam pripada grupi algoritama u vremenskom domenu i kreirao ga je MekLaud [5]. Baziran je na razlici kvadrata koja se koristi umesto proizvoda kod autokorelacije. Algoritmi ove grupe obično zahtevaju neke vidove dodatne obrade signala, bilo da je u pitanju prethodna pripremna obrada (eng. *preprocessing*), ili obrada nakon dobijanja rezultata (eng. *postprocessing*). Kod algoritma MPM to nije slučaj. Ovo omogućava da algoritam bude prilično robustan i da može da detektuje i visoke frekvencije.

Razlika kvadrata se definiše kao:

$$d_t(\tau) = \sum_{i=t+1}^{t+W} (x_i - x_{i+\tau})^2$$

odnosno, kao:

$$d_t^{'}(\tau) = \sum_{i=t+1}^{t+W-\tau} (x_i - x_{i+\tau})^2$$

Postoji veza između autokorelacije i ove funkcije. Ako se raspiše poslednja jednakost, dobija se:

$$d_t^{'}(\tau) = \sum_{i=t+1}^{t+W-\tau} (x_i^2 + x_{i+\tau}^2 - 2x_i x_{i+\tau})$$

Dalje, ako se definiše:

$$m_t^{'}(\tau) = \sum_{i=t+1}^{t+W-\tau} (x_i^2 + x_{i+\tau}^2)$$

može se napisati veza između autokorelacije i razlike kvadrata kao:

$$d_t^{'}(\tau) = m_t^{'}(\tau) - 2r_t^{'}(\tau)$$

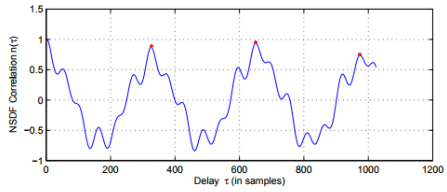
Sledeći korak u algoritmu jeste određivanje vrednosti τ koja odgovara fundamentalnoj frekvenciji. Obično je to neki lokalni minimum funkcije $d_t(\tau)$. Bez opsega vrednosti je teško odrediti kom minimumu odgovara. Normalizovana razlika kvadrata definiše se u algoritmu MPM kao:

$$n_t^{'}(\tau) = 1 - \frac{m_t^{'}(\tau) - 2r_t^{'}(\tau)}{m_t^{'}(\tau)} = \frac{2r_t^{'}(\tau)}{m_t^{'}(\tau)}$$

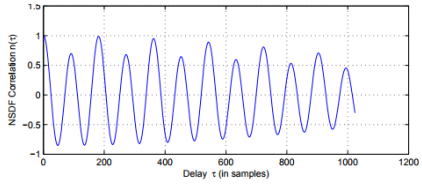
Normalizovana razlika kvadrata $n_t^{'}(\tau)$ je u opsegu od -1 (savršena negativna korelacija) do 1 (savršena korelacija).

Sledeći korak algoritam je nalaženje vrha (eng. *peak picking*). Da bi se on odredio, uzima se najveći maksimum između svakog pozitivnog i negativnog preseka sa x osom. Ovi maksimumi se nazivaju "ključni maksimumi". Na slici 2.2 su prikazana tri takva maksimuma. Napravi se prag tako što se uzme najveći maksimum i pomoži sa konstantom *k*, i uzima se pomeraj τ prvog maksimuma koji je veći od ovog praga za period. Konstanta *k* mora da bude dovoljno velika da se izbegne biranje onih vrhova koji odgovaraju jakim harmonicima (slika 2.3), ali i dovoljno malo da se ne izabere neželjeni pod-harmonik. U praksi je to najčešće vrednost između 0.8 i 1.

Postoje još neki algoritmi koji koriste razliku kvadrata umesto autokorelacije od kojih je najpoznatiji Jinov algoritam [3].



Slika 2.2: Primer grafika normalizovane razlike kvadrata

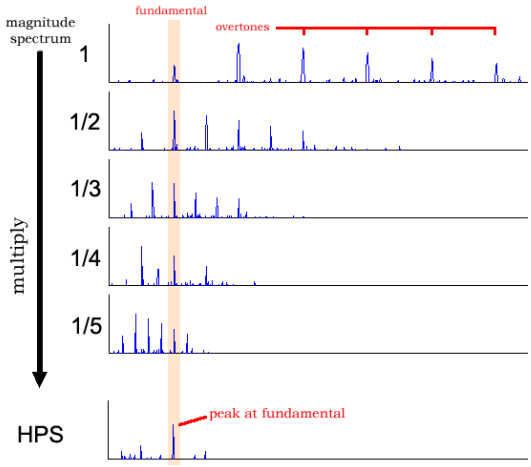


Slika 2.3: Slučaj lošeg biranja konstante k. Ovde je pravi period na 190 (ili 84.21Hz), ali su moguća biranja i na pola od ove vrednosti.

2.1.3 Analiza spektra

Na slici 1.1 prikazan je spektar tona C odsviranog na klaviru. Kada se pređe iz vremenskog domena u frekvencijski domen, dobija se spektar signala koji oslikava frekvencije koje se nalaze u signalu, u ovom slučaju u zvučnom signalu. Prelazak iz vremenskog domena u frekvencijski može se izvesti korišćenjem Furijeovih transformacija.

Mnogi algoritmi za nalaženje fundamentalne frekvencije rade u frekvencijskom domenu i počivaju upravo na analizi spektra. Jedan od tih algoritama je spektar proizvoda harmonika (eng. *harmonic product spectrum*), u daljem tekstu HPS.



Slika 2.4: Ilustracija HPS algoritma

Algoritam je prilično jednostavan. Signal se podeli na delove i onda se svaki deo prevede u frekvencijski domen primenom brze Furijeove transformacije. Zatim se za svaki deo smanjuje učestalost uzorkovanja (eng. *downsampling*) spektra n puta. Parametar n zavisi od broja harmonika. Zatim se nađe proizvod ovih smanjenih spektara i fundamentalna frekvencija u rezultatu. Na slici 2.4 je dat primer smanjivanja učestalosti uzorkovanja pet puta.

Greške koje nastaju u ovom algoritmu obično su vezane za biranje oktave umesto fundamentalne frekvencije, tako da je ponekad potrebna i dodatna obrada. Algoritam je

dosta brz i efikasan i robustan na šum. Brzina i efikasnost se smanjuje ako se koriste sporije Furijeove transformacije koje daju bolji rezultat.

Popularni algoritam u frekvencijskom domenu je i algoritam Cepstrum [4], koji može da se kombinuje sa algoritmom HPS.

2.1.4 Perceptualni detektor fundamentalne frekvencije

Posebno su interesantni algoritmi koji modeliraju ljudski sistem percepcije zvuka. Ukoliko je modeliranje precizno, dobijeni algoritam je jako robustan, neosetljiv na šum ili distorziju zvuka, budući da ljudi imaju sposobnost da prepoznaju fundamentalnu frekvenciju i kod signala koji nisu periodični ili koji imaju dosta nepravilnosti i nečistoća koje nastaju snimanjem zvuka. Jedan od pristupa su predložili Slejni i Lajon u svom radu [1]. Njihov model ima tri dela:

Model kohlee ili puža

Kohlea (eng. *cohlea*) je organ čula sluha koji je smešten u unutrašnjem uvu. Sastoji se od tri kanala koja su savijena u obliku spirale koja liči na puževu školjku, zbog čega se još zove i puž. Kada zvuk uđe do kohlee, talasi se propagiraju niz bazilarnu membranu koja odjekuje u svakom trenutku najsnažnije talasom određene frekvencije. Do ovoga dolazi zato što se ukočenost bazilarne membrane menja po njenoj dužini, i na različitim mestima utiču talasi različitih frekvencija. Dakle, svaki deo kohlee reaguje najbolje na neku drugu frekvenciju, što se može videti na slici 2.5. Gornji delovi kohlee su osetljiviji na više frekvencije i kako se ide niže niz kohleu detektuju se niže frekvencije. Na ovaj način talas je podeljen po frekvencijama u kohlei [19]. U ovom modelu, ovaj deo je modeliran filterima drugog reda.

Pokrete membrane detektuju ćelije dlačica. Ovo je modelirano pomoću polutalasnih ispravljača (eng. *Half Wave Rectifier*) koji kao ulaz imaju izlaz iz prethodnih filtera koji dalje konvertuju kretanja membrane u signal koji sadrži i ovojnicu i vremensku strukturu.

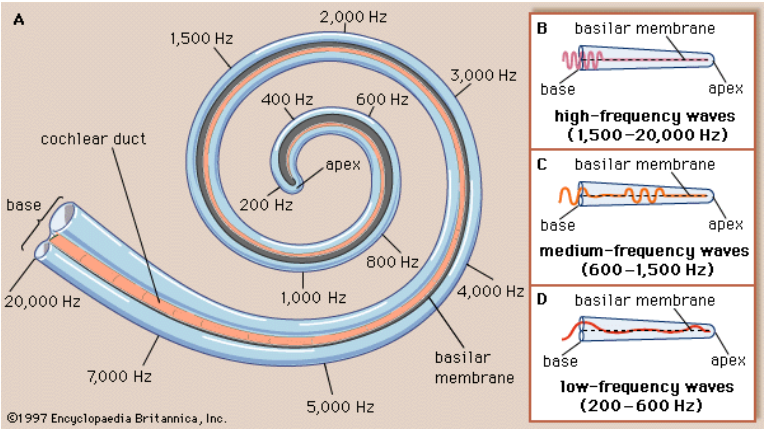
Kasnije se signal prevodi u signal koji može da se šalje preko nerava mozgu. Ovo je modelirano korišćenjem četiri nivoa automatske kontrole jačine (eng. *Automatic Gain Controll*).

Korelogram

Korelogram je animirana slika zvuka koja ima frekvencijske sadržaj po vertikalnoj, a vremensku strukturu po horizontalnoj osi. On se dobija računanjem autokorelacije izlaza iz svakog kanala kohlee i predstavlja dobar metod za nalaženje perioda kod periodničnih signala. U ovom slučaju, budući da se vrhovi u svim kanalima dešavaju u istom pomeraju na slici korelograma formira se vertikalna linija.

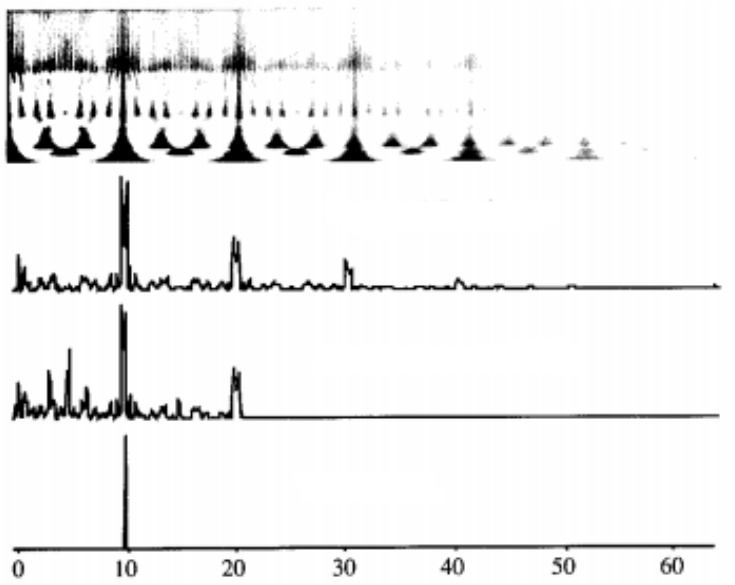
Detektovanje fundamentalne frekvencije pomoću korelograma ima 3 podkoraka, a ilustracija je data na slici 2.6:

- Preprocesiranje od dva nivoa. U prvom se naglašavaju vertikalne linije u korelogramu, a u drugom oni vrhovi koji pokazuju periode u signalu.



Slika 2.5: Kohlea i frekvencije

- Sumiranje vrednosti po svim frekvencijama u svakom vremenskom pomeraju. Ovo daje mogućnost procenjivanja svih potencijalnih perioda u korelogramu.
- Bira se najveći vrh, vodeći računa o greškama oktave.



Slika 2.6: Ilustracija detektovanja fundamentalne frekvencije pomoću korelograma

2.2 Detektovanje dužina tonova

Dužina tona je jako bitan parametar svake transkripcije. Da bi se detektovala dužina najčešće je potrebno pronaći početak (eng. *onset*) i kraj (eng. *offset*) tona. Ponekad posle kraja tona počinje drugi ili sledi pauza (eng. *rest*), tako da je potrebno detektovati pored nota i dužina i pauze i njihove dužine. Postoji više načina da se detektuje dužina tona, a ovde će biti prikazana dva.

2.3 Ovojnica zvuka

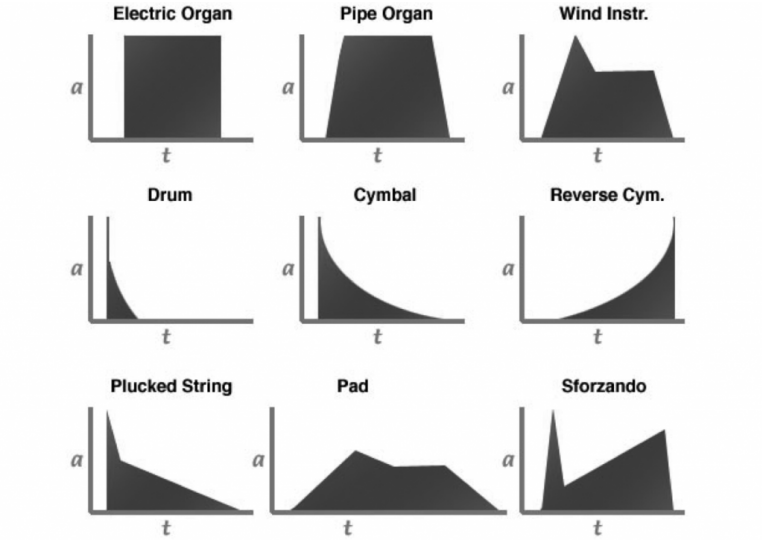
Ovojnica (eng. *envelope*) zvuka pokazuje kako se amplituda menja tokom zvuka, odnosno kako zvuk postaje glasniji ili tiši tokom vremena. Jedan od modela ovojnica je ADSR ovojnica i njeni elementi su:

- Napad (eng. *attack*) koji predstavlja vreme koje prođe od nastanka zvuka do dostizanja pune amplitude.
- Opadanje (eng. *decay*) koje predstavlja vreme koje prođe od pune amplitude do nivoa zadržavanje zvuka.
- Zadržavanje (eng. *sustain*) koje predstavlja vreme kada se amplituda stabilizuje.
- Otpuštanje (eng. *release*) koje predstavlja vreme koje je potrebno da zvuk pređe iz nivoa zadržavanja u tišinu.

Ovojnica je bitan element boje zvuka i određuje dinamiku signala. Tako različiti instrumenti proizvode zvukove koji imaju različite ovojnice jer su ADSR elementi specifični za svaki instrument. Npr. delovi bubnjeva imaju jako kratko (skoro neprimetno) vreme napada (kao i ostale delove ovojnice), dok je kod violine to vreme (i ovojnica) obično mnogo duže. Ovojnice različitih instrumenata date su na slici 2.7. Dobijanjem ovojnice mogu se dalje dobiti bitne informacije o zvuku, kao što su informacije o amplitudi i trajanju.

2.3.1 Detektovanje trajanja na osnovu detektovanih nota

Ovaj način detekovanja trajanja predložio je Monti [2]. Ako se digitalni signal obrađuje bafer po bafer i ako je veličina bafera 1024 uzoraka, a učestalost uzorkovanja 44,1KHz, onda jedan bafer predstavlja 23,21ms ($\frac{1024}{44100*1000}$). Kod ovog pristupa jako je bitno dati kao parametar minimalno trajanje tona. Kada detektovani tonovi održavaju istu vrednost duže od minimalnog trajanja onda je pronađen i prepoznat početak tona. Ako je niz detektovanih tonova kraći od ovog parametra onda se ceo niz odbacuje. Ukoliko se u tom prozoru koji određuje minimalno trajanje detektovani tonovi promene ali ga ne premaše, i prethodno detektovani ton bude ponovo prepoznat, onda se i taj prozor uzima kao deo detektovanog tona. Na ovaj način se parametrom određuje i memorija sistema. Ako se amplituda podigne više puta u nizu istih detektovanih tonova znači da su isti tonovi odsvirani više puta jedan za drugim, pa i o tome treba voditi računa. Takođe je zgodno izračunati energiju signala u baferu ili ovojnicu i na taj način prepoznavati tišinu, odnosno pauzu.



Slika 2.7: Ovojnice različitih intrumenata

Kraj tona je detektovan kada se detektuje pauza ili kada se detektuje novi ton. Kada je poznat početak i kraj tona, njihovom razlikom se lako dobija trajanje tona.

Poglavlje 3

Implementacija

Uz ovaj rad razvijena je aplikacija za automatsku transkripciju monofonih melodija. Aplikacija je napisana korišćenjem radnog okvira (eng. *framework*) .NET i programskog jezika C♯. Za obradu zvuka korišćena je biblioteka NAudio. Izlaz iz aplikacije je u formatu MusicXML. U ovom poglavlju biće opisani detalji implementacije i pregled tehnologija koje su pomogle razvoj programa.

3.1 Biblioteka NAudio

NAudio [9] je .NET biblioteka otvorenog koda. Napisana je od strane Marka Hita 2002. godine usled nedostataka u radnom okviru .NET za manipulaciju zvukom. Sadrži veliki broj klasa i omotača (eng. *wrapper*) koje olakšavaju razvoj i odžavanje audio aplikacija.

Glavna mana biblioteke NAudio dolazi iz ograničenja koje proizilaze iz jezika i radnog okvira, što se uglavnom odražava na neregularnosti kada se radi sa niskom latentnošću, zbog toga što .NET koristi sakupljač smeća (eng. *garbage collector*) koji može da prekine proces u bilo kom trenutku, što se kod zvuka manifestuje kao prekid (eng. *glitch*). Latentnost je kratak vremenski period kašnjenja između ulaska i izlaska zvuka iz nekog sistema. Međutim, ovaj problem ne postoji kod konkretne aplikacije tako da se NAudio pokazao kao dobar izbor.

Isto tako, NAudio je razvijan tako da je smanjena aktivnost sakupljača smeća, pa su u velikoj meri ublaženi mnogi problemi koje on može da izazove prilikom razvoja zvučnih aplikacija.

NAudio sadrži:

- omotače za audio API-eve (eng. *application programming interface*),
- podršku za čitanje i pisanje za najpopularnije formate,
- interfejse za konstrukciju lanaca signala (eng. *signal chains*),
- korisne audio klase i komponente lanaca signala.

Upotrebom biblioteke NAudio olakšan je rad sa kodecima, snimanje i pokretanje zvuka, implementacija zvučnih efekata, menjanje zvučnih formata, menjanje učestalosti uzorkovanja, pristup i rad sa pojedinačnim uzorcima, rad sa različitim formatima datoteka, audio prenos putem Interneta, itd.

U nastavku su opisane osnovne karakteristike i funkcionalnosti biblioteke NAudio koje su upotrebljene u razvoju aplikacije. Puna dokumentacija se nalazi na zvaničnoj Veb lokaciji [9].

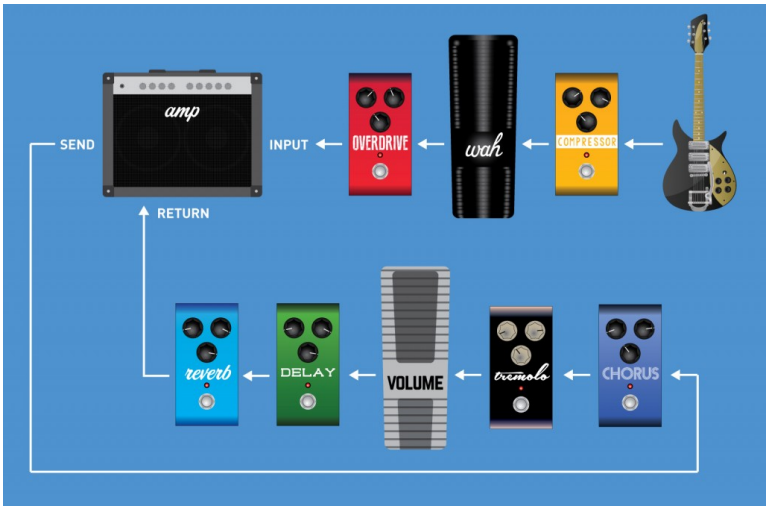
3.1.1 Omotači, formati i lanci signala

NAudio sadrži omotače za mnoge API-je, bilo da su to stariji Vindouz API-evi (waveIn, waveOn, midiIn, midiOut, acm...) ili moderniji (Wasapi, ASIO, Media Foundation API...). Podržani su svi popularniji formati datoteka, među kojima su najbitniji: WAV, MP3, AIFF, MIDI.

Audio lanac signala je put koji zvuk pređe od izvora do izlaza. NAudio omogućava olakšano kreiranje audio lanaca. Kao deo lanca mogu se koristiti:

- ulazi i izlazi (neka datoteka ili zvučna karta),
- kodeci, koji menjaju audio format (dekodiranje u PCM, kodiranje u neki kompresovani format, npr. MP3),
- vizuelizacija, koja omogućava grafički prikaz i analiziranje zvuka,
- efekti, koji modifikuju zvuk (npr. efekat kašnjenja (eng. *delay*)),
- mikseri, koji omogućavaju kombinovanje više ulaza,

  i mnogi drugi.



Slika 3.1: Lanac signala na primeru gitare

3.1.2 Najbitnije funkcionalnosti biblioteke NAudio

WaveStream

Klasa WaveStream je jedna od prvih klasa dodatih u NAudio i nasleđuje System.Stream klasu. Svojstva klase WaveStream su dužina datoteke i pozicija (koja može da se menja), i format datoteke (sadrži informacije o broju bitova po uzorku, učestalosti uzorkovanja, broju kanala itd). Najbitniji metod je metod za čitanje (metod Read) koji čita bajtove uzoraka redom u smešta ih u niz (bafer). Na ovaj način omogućen je pristup samim uzorcima i rad sa njima. Pristup uzorcima zajedno sa podacima o formatu daje mogućnost razvijanja funkcionalnosti kao što su promena broja bitova po uzorku, učestalosti uzorkovanja, manipulaciju na nivou bitova, implementiranje efekata i slično.

IWaveProvider i ISampleProvider

IWaveProvider i ISampleProvider su interfejsi koji predstavljaju uprošteniju verziju klase WaveStream jer ne sadrže informacije o dužini i poziciji. IWaveProvider i WaveStream rade sa nizovima bajtova, dok ISampleProvider radi sa nizovima brojeva u pokretnom zarezu (tj. sa *float* tipom podataka), što olakšava manipulaciju i rad sa pojedinačnim uzorcima. NAudio ima mnoge provajdere koji implementiraju ove interfejse, ali prava snaga i fleksibilnost dolazi sa jednostavnošću kojom je moguće implementiranje dodatnih provajdera.

Rad sa datotekama

NAudio ima ugrađene klase za čitanje datoteka koje implementiraju WaveStream. Najbitnije su WaveFileReader, Mp3FileReader, AiffFileReader, WmaFileReader. Svaka od njih ima svoju odgovarajuću klasu za pisanje, odnosno odgovarajuću Writer klasu.

Jako bitna klasa za čitanje je klasa AudioFileReader koja omogućava čitanje većine formata, implementira ISampleProvider i omogućava rad sa float uzorcima.

Postoje mnoge klase za konvertovanje provajdera kao što su SampleToWaveProvider16 ili WaveFloatTo16Provider.

Filteri i FFT

NAudio implementira filtere kao što su:

- niskopropusni filter (eng. *low pass filter*) koji propušta samo frekvencije niže od nekog praga,
- visokopropusni filter (eng. *high pass filter*) koji propušta samo frekvencije više od nekog praga,
- pojasnopropusni filter (eng. *band pass filter*) koji propušta samo frekvencije u nekom opsegu,
- propusni filter sa donjom granicom (eng. *low shelf filter*) koji propušta sve frekvencije ali povećavanja ili smanjuje frekvencije niže od nekog praga,

- propusni filter sa gornjom granicom (eng. *high shelf filter*) koji propušta sve frekvencije ali povećavanja ili smanjuje frekvencije više od nekog praga.

Postoji i podrška za brzu Furijeovu transformaciju i prozorske funkcije (eng. *windowing function*) kao što su Hamingova i Hanova prozorska funkcija. Curenje spektra (eng. *spectral leakage*) koje se javlja kod FFT-a se ublažava tako što se signal u vremenskom domenu množi prozorskom funkcijom.

3.2 MusicXML

MusicXML je XML format za razmenu i distribuciju digitalnih partitura (eng. *sheet music*) [15]. Koriste ga mnogi muzički programi i baze podataka. Nudi dosta mogućnosti za zapis najrazličitijih elemenata muzike. Ovde će biti naveden podskup koji je dovoljan za zapis monofonih melodija.

3.2.1 Struktura MusicXML datoteka

Ako muzika ima više delova (eng. *part*) i više taktova (eng. *measure*) postoje dva načina da se organizuje i zapiše u MusicXML-u. Ukoliko su delovi primarni, a taktovi sadržani u njima, glavni element (eng. *root element*) je `<score-partwise>` koji sadrži više `<part>` elemenata, a oni dalje sadrže više `<measure>` elemenata. Ako je obrnuto, glavni element je `<score-timewise>` koji sadrži `<measure>` elemente a oni `<part>` elemente. Konvertovanje iz jednog u drugi tip datoteke je jednostavno.

Svaka MusicXML datoteka sadrži *score-header* entitet koji sadrži neke osnovne metapodatke o muzičkom delu i listu delova na koje je to delo podeljeno. Na nivou elemenata, jedino je `<part-list>` element koji sadrži delove, odnosno `<score-part>` elemente, obavezan. Ostali elementi koji nisu obavezni su: `<work>`, `<movement-number>`, `<movement-title>` i koriste se za zapis stavova (eng. *movements*). MusicXML datoteka može da sadrži i `<identification>` element koji sadrži metapodatke o muzičkom delu.

Najjednostavnija MusicXML datoteka je data u nastavku. U pitanju je jedan takt koji sadrži samo jednu notu (srednje C), dok je dužina takta 4/4.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
    "-//Recordare//DTD MusicXML 3.0 Partwise//EN"
    "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
```

```
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

Postoje dve vrste elemenata u fomatu MusicXML. Jedna grupa definiše kako muzika zvuči, a druga kako se zapisuje. Na osnovu prve grupe moguće je kreirati MIDI datoteku, dok je na osnovu druge grupe moguće napraviti partituru. U sledećoj sekciji je dat kratak pregled elemenata koji su relevantni u ovom radu za transkricipiju monofonih melodija. To su elementi iz prve grupe.

3.2.2 Neki MusicXML elementi

Na početku MusicXML datoteke je obično `<attributes>` element koji između ostalih sadrži sledeće elemente:

- `<divisions>` element kojim se navodi broj divizija u četvrtini note, i na osnovu ovog broja mogu se napisati trajanja ostalih nota. Ako je sadržaj elementa 24, onda je trajanje osmina nota 12 a šesnaestina 6.
- `<time>` element koji sadrži elemente za zapis tempa: `<beats>` i `<beats-type>`. Ako su oni, redom, 7 i 8, onda govorimo o tempu 7/8.

Sledeći bitan element je `<note>` element koji sadrži sledeće elemente:

- `<pitch>` element koji označava visinu tona. Sadrži elemente `<step>` koji sadrži ime note u abecednoj formi, `<alter>` element koji može biti -1 ili 1 u zavisnosti

od toga da li je ton povišen ili snižen, a informacija o broju oktave sadržana je u `<octave>` elementu.

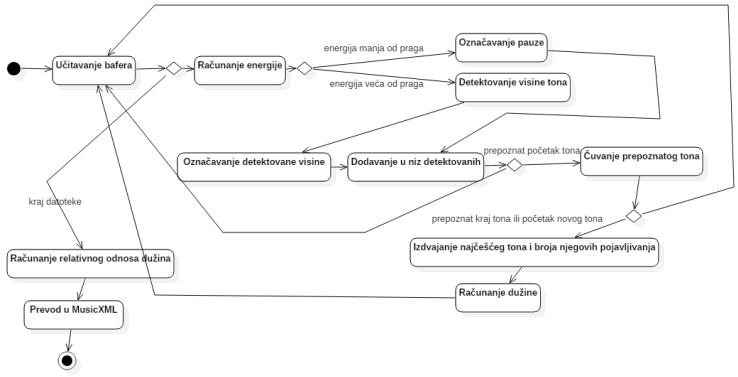
- `<duration>` element koji sadrži trajanje nota u odnosu na referentni broj koji je kao trajanje četvrtine note naveden u `<divisions>` elementu.

3.3 Implementacija

Postoje mnogobrojna rešenja koja nude detektovanje visine tonova, automatsku muzičku transkripciju i slične obrade zvučnog signala koje su u vezi sa ovim radom. Neka od njih su:

- Chordify [8] koji nudi automatsko prevođenje pesme u akorde.
- Tony [22] koji nudi detektovanje visina tonova, transkripciju, razne vizuelizacije.
- Tartini [21] koji nudi razne analize, konture visine tonova, razne vizuelizacije.
- Transcribe [23] koji umnogome olakšava transkripciju tako što omogućava lako usporavanje i ponavljanje cele pesme ili nekih delova, integrisani editor.
- Audacity [7] koji nudi snimanje i editovanje sa velikim brojem mogućnosti i efekata.

Transkripcija monofonih melodija i ljudskog glasa zahvaljujući algoritmima koji su opisani u prethodnom poglavlju predstavlja problem koji je moguće rešiti u zadovoljavajućoj meri. U nastavku su detalji implementacije automatske transkripcije koja je urađena u sklopu ovog rada.



Slika 3.2: Dijagram aktivnosti sistema za automatsku transkripciju monofonih melodija implementiranog u sklopu ovog rada.

3.3.1 Detektovanje visine tonova

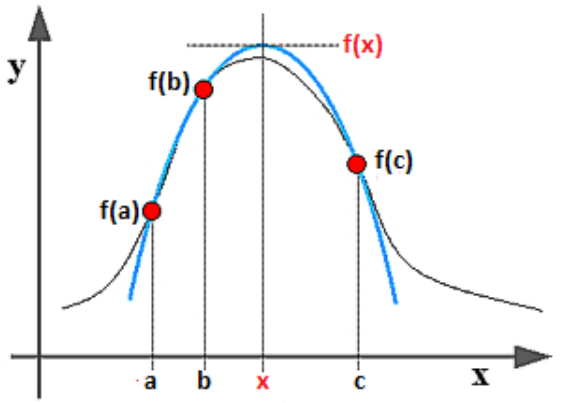
Za detektovanje visine tonova odabran je algoritam MPM opisan u poglavlju 2.1.2 zbog svoje robusnosti, preciznosti i odličnih rezultata. Još jedna od pogodnosti je što algoritam ne zahteva dodatne obrade i upotrebu filtera. MekLeod preporučuje rad sa baferima dužine 512, 1024, 2048 i 4096 uzoraka, i preklapanjem bafera od 75% [5]. Takođe, predloženo je ubrzanje autokorelacije tako što se prvo primenjuje brza Furijeova transformacija nad signalom, zatim pomnoži svaki kompleksni koeficijent svojim konjugatom i na kraju primeni inverzna Furijeova transformacija nad dobijenim međurezultatom.

Za preciznije nalaženje maksimuma upotrebljena je parabolička interpolacija uzimanjem najveće lokalne vrednosti i dva suseda, jednog sa leve, i jednog sa desne strane. Neka su takva tri susedna uzorka a, b i c, $f(a)$, $f(b)$ i $f(c)$ vrednosti funkcije razlike kvadrata u tim tačkama, a x tačka u kojoj je maksimum $y = f(x)$, kao što je na slici 3.3. Ako su a, b i c susedni uzorci, onda važi da je:

$$b - a = c - b = 1$$

Vrednost x će biti blizu tačke b, pa je formula za kvadratnu aproksimaciju:

$$f(x) \approx f(b) + f^{'}(b)(x - b) + \frac{f^{''}(b)}{2}(x - b)^2$$



Slika 3.3: Parabolička aproksimacija

Iz uslova da parbola mora da prođe kroz ove tačke dalje sledi:

$$f^{''}(b) = 2\frac{\frac{f(c)-f(b)}{c-b} - \frac{f(a)-f(b)}{a-b}}{c - a}$$

$$f^{'}(b) = \frac{f(c) - f(b)}{c - b} - \frac{f^{''}(b)}{2}(c - b)$$

Pošto su a, b i c susedni uzorci sledi da je:

$$f^{''}(b) = f(c) - 2f(b) + f(a)$$

$$f^{'}(b) = \frac{f(c) - f(a)}{2}$$

Tražena tačka je:

$$x = b - \frac{f^{'}(b)}{f^{''}(b)}$$

$$y = f(b) - \frac{1}{2} f^{'}(b) \frac{f^{'}(b)}{f^{''}(b)}$$

Posle odgovarajućih zamena i sređivanja dobija se:

$$x = b - \frac{f(c) - f(a)}{2(f(c) - 2f(b) + f(a))}$$

$$y = f(b) - \frac{(f(c) - f(a)^2}{8(f(c) - 2f(b) + f(a))}$$

3.3.2 Detektovanje trajanja tonova

Dva načina su razmatrana tokom implementacije za detektovanje trajanja tonova, odnosno njihovih početaka i krajeva. Prvi način jeste detektovanje cele ovojnice. Drugi način opisan je u poglavlju 2.3.1 i baziran je na detektovanim notama.

Detektovanje ADSR ovojnice

Jedan od načina za detektovanje ADSR ovojnice opisao je Koblenski na svom blogu [13]. Proces se sastoji od tri koraka koji predstavljaju jednostavne operacije sa signalom:

- Uklanjanje DC komponente

  DC komponenta zvučnog signala zapravo predstavlja njegov vertikalni pomeraj, tj. pomera signal gore ili dole po y osi. Uklanjanje ove komponente je važno jer uprošćava dalje procesiranje i može se postići oduzimanjem trenutne srednje vrednosti od svakog uzorka redom. Na ovaj način signal se pomera tako da bude centriran oko x ose.

- Apsolutna vrednost signala

  Pošto je posle prethodnog koraka signal centriran oko x ose, sledeći korak je uzimanje apsolutne vrednosti signala čime se negativne vrednosi brišu i ostaje signal samo sa pozitivne strane x ose.

- Uklanjanje vrhova

  Signal obično ima dosta vrhova ili talasa koje je poželjno ukloniti da bi se dobila glatka ovojnica. Ovo se postiže još jednim izračunavanjem prosečne vrednosti i oduzimanjem od uzoraka.

Srednja vrednost koja je pogodna za gore navedene operacije je eksponencijalna srednja vrednost i da bi se izračunala dovoljno je znati trenutni uzorak i prethodnu srednju vrednost:

$$m_i = ws_i + (1 - w)m_{i-1}$$

Ovde je m_i i-ta srednja vrednosti, s_i i-ti uzorak a w težinski koeficijent koji označava uticaj trenutnog uzorka na srednju vrednost i može imati vrednost između 0 i 1.

Detektovanje trajanja zasnovano na detektovanim visinama tonova

Detektovanje trajanja tonova na ovaj način objašnjeno je u poglavlju 2.3.1. Glavni razlog zbog kojeg je odabran ovaj pristup jeste zbog toga što NAudio pruža mogućnost pristup baferima uzoraka redom, što u potpunosti odgovara predloženom algoritmu.

3.4 Ostali detalji implementacije

3.4.1 Računanje energije

Računanje energije u baferu je jako bitno u ovom sistemu. Očigledan razlog jeste taj što omogućava prepoznavanje tišine, a samim tim i pauza. Drugi razlog jeste taj što ukoliko se bafer prepozna kao tišina detektovanje visine tona u tom baferu nije potrebno, pa se samim tim eliminišu suvišna izračunavanja. Za izražavanje energije signala korišćen je RMS (eng. *root mean square*) nivo signala. RMS vrednost bafera, tj. diskretnog signala x_N, se dobija računanjem kvadratnog korena srednje vrednosti kvadrata amplituda u baferu, odnosno:

$$x_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

3.4.2 Relativne dužine tonova

Kao rezultat detektora dužina dobijaju se dužine detektovanih tonova izražene u milisekundama. Pošto standard MusicXML zahteva da se navede referentna dužina note u vidu nekog broja na početku datoteke preko koje se izražavaju sve ostale dužine, u ovom radu upotrebljena je sledeća strategija. Prvo se izračunaju trajanja svih tonova i pauza u milisekundama. Potom se odabere najmanja dužina i preko nje se, uz odgovarajuća zaokruživanja, izraze svi ostali tonovi.

Nedovoljno precizna detekcija trajanja četvrtine note je najozbiljnije ograničenje ovog sistema. Međutim, precizno detektovanje četvrtine note bi zahtevalo detektovanje broja otkacaja u minuti, što umnogome prevazilazi opsege ovog rada. Ukoliko je npr. broj otkucaja u minuti 60 onda u savršenom slučaju četvrtina note traje 1000ms, tj. 1s.

3.4.3 Parametri sistema

Na kvalitet zvučnog zapisa monofonih melodija utiče mnogo faktora, kao što su npr. kvalitet snimanja ili količina prisutnog šuma. Takođe, brzina melodije, jačina tonova i

slični parametri utiču na dinamiku zvuka. Svi ovi, a i mnogi drugi razlozi dovode do toga da sistem za transkripciju mora da ima određene parametre pomoću kojih se prilagođava različitim tipovima melodija i različitim kvalitetima snimaka. U ovom sistemu parametri su minimalno trajanje tona, minimalna jačina tona, minimalna jačina tona u nizu ponovljenih tonova i procenat preklapanja bafera.

Minimalno trajanje tona je parametar koji određuje minimalnu dužinu koju ton treba da održi bez promene vrednosti da bi bio prihvaćen. Uz to, pomaže sistemu da se prilagodi melodijama različite brzine.

Minimalna jačina tona je parametar koji određuje kolika je minimalna vrednost amplitude potrebna da ton ne bi bio prepoznat kao tišina ili pauza.

Minimalna jačina tona u nizu ponovljenih tonova je parametar koji određuje minimalnu vrednost skokova amplitude u nizu ponovljenih istih tonova da bi se oni prepoznali kao dva ili više tonova umesto kao jedan duži ton.

Procenat preklapanja bafera je parametar koji određuje u kojoj meri se preklapaju baferi uzoraka koji se prosleđuju sistemu jedan za drugim.

Poglavlje 4

Diskusija

4.1 Evaluacija implementiranog rešenja

Da bi se izrazio kvalitet implementiranog sistema potrebne su monofone melodije koje sistem pokušava da transkribuje i kriterijum za meru kvaliteta, odnosno izražavanje greške. Budući da je izlaz iz programa niz nota, za računanje greške korišćen je metod koji je predložio Mur u svom radu [18] u kome se porede note u referentnim melodijama sa transkribovanim, i obrnuto, i računa prosek razlika. Tada se greška u procentima može izraziti formulom:

$$E_n = \frac{1}{2}\left[\frac{C_R - C_R^{'}}{C_R} + \frac{C_T - C_T^{'}}{C_T}\right]100$$

Ovde je sa C_R i C_T označen ukupan broj nota u referentnoj, odnosno transkribovanoj melodiji. Sa $C_R^{'}$ označen je broj nota u referentnoj melodiji za koje je sistem prepoznao vreme i tačnu frekvenciju, a sa $C_T^{'}$ obrnuto, broj nota u transkribovanoj melodiji koje se poklapaju sa odgovarajućim notama u referentnoj. Ova greška se računa za svaku referentnu melodiju, a kao konačan rezultat se uzima prosek dobijenih vrednosti.

4.2 Analiza

Korpus referentnih melodija je biran tako da sadrži primere melodija koje imaju reprezentativne karakteristike. Bitne karakteristike koje su uzete u obzir su dužina melodije, različite dužine tonova u melodiji, uzastopno ponavljanje tonova i slične. Ovde će biti reči o rezultatima koji su dobijeni na slučaju šest melodija od kojih su tri svirane na klaviru, dve na violini i jedna na gitari, ali i o uočenim karakteristikama sistema prilikom testiranja na većem korpusu melodija i tokom razvoja. Radi jednostavnosti, testirane su sporije melodije, pošto je usporavanje melodija moguće i vrlo jednostavno upotrebom nekog od dostupnih alata (npr. gore pomenuti Audacity). Melodije su preuzete sa Veba i nisu savršenog kvaliteta, ali pošto je to jedan od glavnih slučajeva upotrebe ovog sistema,

one nisu dodatno obrađivane ili menjane. Prilikom računanja grešaka dužine tonova su uzimane sa određenom tolerancijom, iz razloga koji su opisani u sekciji 3.4.2.

Treba napomenuti da je sistem testiran na dosta melodija, od kojih su mnoge dužine nekoliko tonova ili imaju neke posebne karakteristike (pauze, uzastopna ponavljanja istih tonova i slično). Takve melodije su uglavnom korišćene za izgradnju sistema korišćenjem testova jedinica koda (eng. *unit test*) i tehnike razvoja vođenog testovima (eng. *test driven development*), pa one nisu posebno razmatrane u rezultatu budući da sistem u ovim slučajevima daje odlične rezultate.

Što se parametara tiče, u ovim slučajevima se pokazalo da preklapanje bafera nema uticaja na rezultat, a prilično usporava rad sistema pošto se u tom slučaju obrađuje mnogo više bafera. Za minimalno trajanje tonova uzeta je vrednost od 50ms, a za veličinu bafera 1024 uzoraka. Učestalost uzorkovanja je 44,1Khz, dok je format datoteka WAV. Ostali parametri su prilagođavani konkretnim melodijama.

U nastavku je dat prikaz nekih karakterističnih grešaka sistema prilikom testiranja.

Greške oktave

Iako ova greška ne nastaje često treba je navesti pošto je problematična za mnoge sisteme za automatsku transkripciju. Sistem u ovom slučaju prepoznaje korektan ton ali pogrešnu oktavu na početku i kasnije se ispravlja. Tako npr. ako sistem treba da prepozna ton F3 u trajanju od 560ms, a umesto toga prepozna ton F2 u trajanju od 60ms i nakon toga F3 u trajanju od 500ms, tada sistem pravi grešku oktave. Ova greška se vezuje za detektor visine tonova.

Neprepoznat ton

Ova greška takođe nastaje zato što detektor visine tonova ne prepozna ton, međutim kvalitet snimaka i šum takođe utiču na nastanak ove greške. Ali, tokom testiranja je primećeno da sistem ili prepozna korektan ton ili ne prepozna nijedan ton, dakle nekorektno prepoznavanje tonova je retko primećeno.

Nekorektno prepoznati uzastopni tonovi

Ova greška vezana je za detektor trajanja dužine. Sistem je implementiran tako da za svaki detektovani ton proveri da li je u pitanju jedan duži ton ili nekoliko istih tonova odsviranih jedan za drugim. Ovo je rešeno traženjem vrhova u ovim intervalima. Međutim, kako signali nisu savršeni vrhovi su pronađeni i na dodatnim mestima. Kontrolom parametara ova greška uglavnom može da se izbegne. Otežavajuća okolnost je ta što je teško naći odgovarajuće parametre zbog nesavršenosti signala.

4.3 Greška

U tabeli je dat prikaz melodija i dobijenih grešaka. Ukupna greška sistema na ovom korpusu je 12,09%.

Instrument	Broj tonova	E_n
violina	33	12,87%
violina	38	10,52%
klavir	18	5,55%
klavir	50	11,25%
klavir	22	16,62%
gitara	19	15,78%

Tabela 4.1: Korpus i dobijena greška

4.4 Dalji rad

Potencijalna poboljšanja sistema leže u promeni detektora visine tona i upoređivanju rezultata sa trenutnim. Iako algoritmi koji rade u vremenskom domenu daju dobre rezultate kod monofonih melodija, bilo bi dobro uporediti ih sa algoritmima iz drugih grupa. Kao što je moguće zameniti detektor visine tonova i uporediti rezultate, moguće je i isprobati ponašanje sistema u slučaju da koristi drugi način za detektovanje dužina, npr. ovojnicu ADSR. Takođe, razmatranje proširivanja sistema u smislu dodavanja mogućnosti za transkripciju polifonih melodija i njegovog potencijala za rešavanje ovog problema je odličan sledeći korak u istraživanju i radu.

Poglavlje 5

Zaključak

U radu su date teorijske osnove i praktične smernice koje su potrebne za razumevanje i razvoj sistema za transkripciju monofonih melodija. Isto tako, dat je i pregled osnovnih pojmova iz muzičke teorije koji su potrebni za razumevanje procesa muzičke transkripcije. Takođe, bilo je reči o percepciji zvučnih signala i karakteristikama koje su potrebne da se zvuk opiše u kontekstu muzičke transkripcije. Pored toga, opisane su i tehnike obrade digitalnog signala, što može da posluži kao dobra osnova za razvoj zvučnih aplikacija. U sklopu rada implementiran je sistem za transkripciju monofonih melodija pomoću biblioteke NAudio i radnog okvira .NET, ali su, takođe, dati i alternativni pristupi i metodi osim onih korišćenih u realizaciji. Na kraju je prikazana analiza sistema i predložene su mogućnosti za njegovo poboljšavanje. U radu su opisane i smernice za dalji rad koje ostavljaju veliki broj mogućnosti za različita zanimljiva istraživanja u ovoj oblasti.

Literatura

[1] Malcolm Slayne. “A Perceptual pitch detector”. In: (1990). URL: http://www.slaney.org/malcolm/apple/Slaney1990(PerceptualPitch).pdf.

[2] Monti Giuliano. “Monophonic transcription with autocorrelation”. In: (2000). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.419.1373&rep=rep1&type=pdf.

[3] Alan de Cheveigne and Hideki Kawahara. “YIN, a fundamental frequency estimator for speech and music”. In: (2002). URL: http://audition.ens.fr/adc/pdf/2002_JASA_YIN.pdf.

[4] “Cepstral signal analysis for pitch detection”. In: (2005). URL: http://kom.aau.dk/group/04gr742/pdf/cepstrum.pdf.

[5] Phillip McLeod and Geoff Wyvill. “A smarter way to find pitch”. In: (2005). URL: http://miracle.otago.ac.nz/tartini/papers/A_Smarter_Way_to_Find_Pitch.pdf.

[6] Justin Colletti. “The Science of Sample Rates (When Higher Is Better — And When It Isn’t)”. In: (2013). URL: http://www.trustmeimascientist.com/2013/02/04/the-science-of-sample-rates-when-higher-is-better-and-when-it-isnt/.

[7] *Audacity*. URL: http://www.audacityteam.org/ (visited on 08/14/2017).

[8] *Chordify*. URL: https://chordify.net/ (visited on 08/20/2017).

[9] Mark Heath. *NAudio API*. URL: https://naudio.codeplex.com (visited on 08/20/2017).

[10] Simon Horsey. *Essential music theory*. URL: http://www.essential-music-theory.com/rhythm-tree.html (visited on 04/14/2017).

[11] *ISO 16:1975*. URL: https://www.iso.org/standard/3601.html (visited on 09/11/2017).

[12] Christophe Kalenzaga. *How Shazam works*. URL: http://coding-geek.com/how-shazam-works/ (visited on 03/16/2017).

[13] Sam Koblenski. *Signal Envelopes*. URL: http://sam-koblenski.blogspot.rs/2015/10/everyday-dsp-for-programmers-signal.html.

[14] Topher Lee. *Wav format tutorial*. URL: http://www.topherlee.com/software/pcm-tut-wavformat.html (visited on 04/12/2017).

[15] MakeMusic. *MusicXML*. URL: http://www.musicxml.com (visited on 08/25/2017).

[16] R.F Moore. *Elements of Computer Music*.

[17] Gabriel Chacón Ruiz. *Lossless vs Lossy*. URL: https://cdvsmp3.wordpress.com/cd-vs-itunes-plus-blind-test-results/.

[18] Matti Ryynanen. *Probabilistic modeling of note events in the transcription of monophonic melodies*. URL: http://www.cs.tut.fi/sgn/arg/matti/mryynane_thesis.pdf.

[19] Malcolm Slayne. *Lyon's Cochlear Model*. URL: https://engineering.purdue.edu/~malcolm/apple/tr13/LyonsCochlea.pdf.

[20] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. URL: http://www.dspguide.com.

[21] *Tartini*. URL: http://miracle.otago.ac.nz/tartini/ (visited on 08/16/2017).

[22] *Tony*. URL: https://code.soundsoftware.ac.uk/projects/tony (visited on 07/15/2017).

[23] *Transcribe*. URL: https://transcribe.wreally.com/ (visited on 05/15/2017).

[24] Voxengo. *Voxengo SPAN*. (Visited on 03/16/2017).