Univerzitet u Beogradu

Matematički fakultet Beograd

-Master rad-
Mašinsko prepoznavanje muzičkih uzoraka

autor: Jovan Jovanović
mentor: Miodrag Živković

1. oktobar 2014

Sadržaj

Glava 1

Uvod

Zvuk je mehanička oscilacija čestica nekog medija koje se kroz njega prostiru najčešće kao talas a koje čovek može da čuje. Može se prostirati kroz sva tri agregatna stanja. Grana fizike koja se bavi izučavanjem zvuka naziva se akustika.

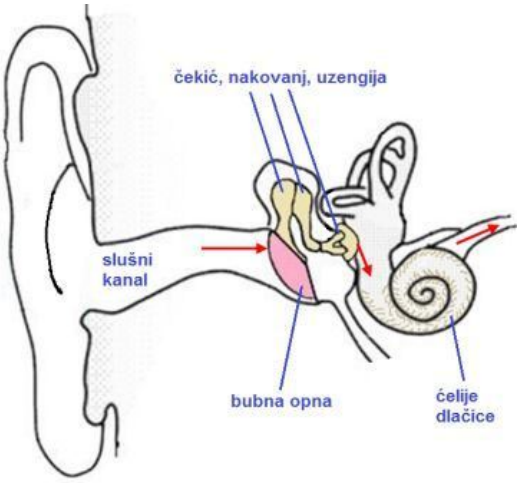
1.1 Kratak uvod u akustiku

Akustika se bavi problematikom generisanja zvuka, njegovim prostiranjem u različitim fizičkim sredinama i efektima koje zvuk izaziva u sredini u kojoj se javlja. U prošlosti akustika je bila samo teorijska grana fizike. Danas je ona veoma široka naučna oblast, koja se proširila na neke teme van inženjerstva. Primer za to su naučne oblasti koje se bave teorijom rada čula sluha, ljudskim govorom i sličnim temama [1]. Akustika se može podeliti na:

• Fizičku akustiku - ispituju se mehaničke pojave u procesima proizvodnje zvuka, njegovo rasprostiranje, širenje elastičnih deformacija u nekoj sredini, efekti refleksije, prelamanja i interferencije.

• Fiziološku akustiku - razmatraju se problemi vezani za organe sluha i govora [2].

Zvučni signal(analogni) je oblik zvuka obično predstavljen električnom strujom [12]. Digitalni signal je niz vrednosti analognog signala. Oblast prepoznavanja zvučnog signala je počela sa razvojem još u doba Aleksandra Grahama Bela. On je pokušavao da razvije uređaj kojim bi opisao zvuk i predstavio ga svojoj gluvoj ženi. Taj uređaj bi u slici prikazivao ulazni zvučni zapis, a gluva osoba bi jednoznačno mogla da tumači dobijenu sliku. Iako je napravio spektografske slike zvuka, njegova žena nije uspevala da ih dešifruje. Posle puno očajanja Bel je ovim putem stigao do otkrića telefona [4].



Slika 1.1: Unutrašnjost ljudskog uha

Zvuk je fizička pojava koja predstavlja sastavni deo čovekovog okruženja, gde se javlja kao prateći element mnogih životnih okolnosti. Zvuk je prisutan gotovo svuda: u samom organizmu čoveka, gde se javlja govor, preko najrazličitih zvukova stalno prisutnih u neposrednom životnom okruženju, pa do zvukova u dubinama okeana ili onih koji u vidu seizmičkih talasa dopiru iz dubine zemlje. Njega koriste neke oblasti umetnosti kao izražajno sredstvo (muzika, film, pozorište, radio, TV), što znači u kreativne svrhe. Zvuk je osnova ljudske komunikacije, pa u raznim oblicima zvukova ljudi nalaze određeni smisao i značenja (govor, razni zvučni signali i slično). Zbog toga se zvukom, osim akustike, na svoj način bave razne društvene nauke i umetnosti.

Zvuk nastaje treperenjem zvučnog izvora. Treperenje se prenosi na susedne čestice u vazduhu koje počinju takođe da trepere i prenose taj efekat na sledeće čestice, itd. Spoljašnje uho kroz *slušni kanal* prenosi talas prema *bubnoj opni*. Bubna opna počinje da vibrira u dodiru sa zvučnim talasima i pokreće lanac sitnih koščica pod nazivom *čekić, nakovanj i uzengija*. Krajnje odredište talasa je unutrašnje uho koje je ispunjeno tečnošću. Kako zvučni talasi dolaze do unutrašnjeg uha tako ta tečnost počinje da se pomera i da istovremeno pokreće sičušne ćelije u obliku *dlačica*. Tako se zvučni talasi preko vibracija i ćelija prevode u poruku, tj. u električne imulpse, proizvedene od strane gore spomenutih *dlačica*, koji se auditornim nervom sprovode do mozga [1]. Na slici 1 predstavljena je gradja ljudskog uha.

CD kvalitet zvuka		
44 kHz	16 bit stereo	10.3 MB
44 kHz	8 bit stereo	5.18 MB
FM radio kvalitet zvuka		
22 kHz	16 bit stereo	5.18 MB
22 kHz	8 bit stereo	2.59 MB
AM radio kvalitet zvuka		
11 kHz	16 bit stereo	2.59 MB
11 kHz	8 bit stereo	1.29 MB

Tabela 1.1: Prikaz veličina zvučnih fajlova u trajanju od jednog minuta

1.2 Formati digitalnog zvučnog signala

Zvučni formati se mogu podeliti u tri glavne kategorije – nekomprimovani format, komprimovani format bez gubitka (loseless) i komprimovani format sa gubicima (lossy). Uz to postoje i različiti 'kodeci' (compressor/decompressor) – programi koji komprimuju, odnosno dekomprimuju zvučne podatke. Nekomprimovani podaci su potpuni u svojoj informaciji, ali i veličini. Primer je format LPCM koji služi za skladištenje muzike na audio CD. LPCM koristi WAV kontejner, gde je WAV windows implementacija RIFF kontejnera. RIFF je generički fajl kontejner koji služi za skladištenje zvuka u komadima. Dodavanjem meta podataka u WAV(RIFF) kontejner dobijamo LPCM format. U WAV formatu se najčešće vrše snimanja za kasniju reprodukciju, obzirom da je ovaj format široko rasprostranjen. U tabeli 1 može se videti odnos utroška prostora na hard disku u odnosu na učestanost i rezoluciju zvuka [10]. Primer komprimovanog podatka s malim gubicima (loseless) je WMA audio format. Ovo je Microsoftov format koji poseduje visok kvalitet. Komprimovani format sa gubicima (lossy) su npr. MP3 i Real Audio format. Ti se formati najčešće i koriste za razmenu na internetu zbog svoje veličine. MP3 format je tip kompresije iz serije MPEG(MPEG-1, MPEG-2, MPEG-3, MPEG-21) koji suzbija vrlo visoke i vrlo niske frekvencije u frekventnom domenu signala. Naime ljudsko uho može čuti u rasponu od 20-40hz u niskoj učestanosti odabiranja i do 20000hz u visokom frekeventom opsegu. Najbolje se čuju zvuci sa frekvencijom oko 1000hz.Ovaj tip kompresije nastoji da sačuva najčujniji spektar, doda određene filtere i pri tom smanji veličinu muzičkog fajla i do 12 puta.

1.3 FFT i primena u obradi zvuka

Frekvencijski spektar nekog signala u vremenskom domenu je reprezentacija tog signala u frekvencijskom domenu. Većina audio signala u primenama pojavljuje se u vremenskom domenu. Ovakav signal je pogodan za praćenje promena signala kroz vreme, ali ne i za digitalnu obradu. Iz tog razloga, korisno je da se taj signal prebaci u frekvencijski domen (frekvencijski spektar) gde postoji tačan podatak koja frekvencija signala se najviše pojavljuje u muzičkom uzorku. Izlazni signali iz pretvarača (senzora) su kontinualni u vremenu, ali računarska obrada zahteva da ti signali budu diskretni. Diskretni signali se generišu tako što se kontinualni signali odabiraju (uzorkuju, mere) u određenim intervalima. Najvažniji zahtev koji odabiranje mora da zadovolji je da se sačuva informacija originalnog signala nakon odabiranja. Odabiranje mora da bude takvo da se digitalna predstava signala može iskoristiti za rekonstrukciju signala iz digitalnog u kontinualni domen. Dva zahteva koje bi trebalo da zadovolji perioda kojom se odabira signal su:

- Originalni signal mora imati ograničen frekvencijski opseg (mora da ima konačan frekvencijski sadržaj),
- Uzorkovanje se radi sa učestanošću odabiranja koja mora biti bar dva puta veća od najveće frekvencije originalnog signala [11]. Ovo tvrdjenje je zapravo Nyquist–Shannon teorema koja je praktično most izmedju kontinualnog i diskretnog signala. Primenljiva je samo na klasu matematičkih funkcija čije su Furijeove transformacije jednake nuli van frekvencijskog opsega.

Za prebacivanje signala u frekvencijski domen koristi se FFT. Brza Furijeova transformacija (engl. Fast Fourier transformation; skraćenica FFT) je algoritam za „brzo“ izračunavanje vrednosti diskretne Furijeove transformacije(definisana u daljem tekstu) [3]. DFT je veoma važna u domenu frekvencijske analize zato što uzima diskretni signal u vremenskom domenu i transformiše taj signal u frekvencijski domen.

Neka je R polje realnih brojeva, u kojem je broj Z jedinica. Neka je, u R w primitivni n-ti koren iz 1, tj $w^n = 1$ i $w^k \neq 1$ za $k < n$, na primer $w = e^{\frac{-j2\pi}{n}}$, gde je j imaginarna jedinica.

Diskretna Furijeova transformacija vektora $x = (x_0, \ldots, x_{n-1})$ je vektor $\hat{x}$ definisan na sledeći način:

$$\hat{x}_k = \sum_{j=0}^{n-1} w^{j \cdot k} \cdot x_j$$

za $k = 0, \ldots, n-1$,

gde je w prethodno definisano. Inverzna diskretna Furijeova transformacija izražava vektor x preko $\hat{x}$ na sledeći način:

$$x_k = \frac{1}{n} \sum_{j=0}^{n-1} w^{-j \cdot k} \cdot \hat{x}_j$$

za $k = 0, \ldots, n-1$. Dakle potreban operacija je $(n-1) \cdot (n-1)$, odnosno kompleksnost DFT algoritma je $O(n^2)$

Postupci za izračunavanje FFT se zasnivaju na razlaganju N članova niza u nekoliko podnizova. Tako se diskretna Fourierova transformacija (DFT) dužine N može izraziti preko dve diskretne Fourierove transformacije dužine N/2, pri čemu jedna od njih sadrži parne, a druga neparne članove ulaznog niza $x(n)$. Osnovno razlaganje za pretpostavku da je $N = 2^k$:

$$X_k = \sum_{m=0}^{\frac{N}{2}-1} X(2m)(W_N^2)^{mk} + W_N^k \cdot \sum_{m=0}^{\frac{N}{2}-1} X(2m+1)(W_N^2)^{mk}$$

.

Znajući da je $W_N^2 = W_{N/2}$ važi:

$$X_k = \sum_{m=0}^{\frac{N}{2}-1} X(2m) W_{N/2}^{mk} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} X(2m+1) W_{N/2}^{mk}$$

.

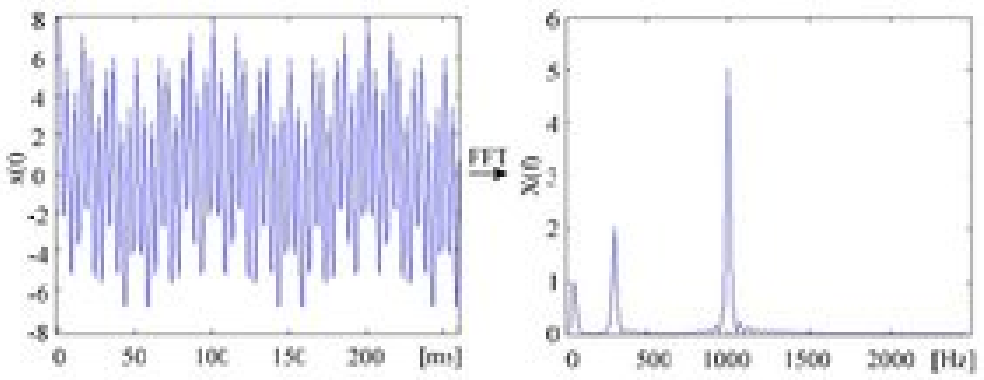
Sama transformacija se sastoji iz parnih i neparnih podnizova:

Neka je $F_{parno} = \sum_{m=0}^{\frac{N}{2}-1} X(2m) W_{N/2}^{mk}$ i $F_{neparno} = \sum_{m=0}^{\frac{N}{2}-1} X(2m+1)(W_N^2)^{mk}$ za $k = 0, 1, ..., N/2 - 1$.

Tada je $X_k = F_{parno}(k) + W_N^k F_{neparno}(k)$

i $X_{k+N/2} = F_{parno}(k) - W_N^k F_{neparno}(k)$ $k = 0, 1, ..., N/2 - 1$.

Broj izračunavanja u slučaju FFT algoritma je znatno manji i kompleksnost iznosi $O(n \cdot logn)$. Algoritam FFT izražava postojeći digitalni signal (def. digitalnog signala spomenuta u 1.1) kao linearnu kombinaciju određenog broja sinusoida. Sa druge strane, ako postoji potreba da se signal



Slika 1.2: Transformacija signala x(k) = cos(2πk·50)+2cos(2πk·300)+5

vrati nazad u vremenski domen, primenjuje se inverzna Furijeova transformacija, koja se takodje efikasno izračunava primenom FFT. Slika 2 predstavlja primer konverzije iz vremenskog u frekvencijski domen funkcije $x(k)$, približno definisane izrazom cos(2πk·50)+2cos(2πk·300)+5.

Furijeova transformacija predstavlja matematičku tehniku za izračunavanje različitih signala, kao što je dinamička promena napona u žici koja povezuje uređaj sa zvučnikom – kao linearnu kombinacija sinusoida učestanosti (frekvencija).

1.4 Prikaz komercijalnog softvera za prepoznavanje muzičkih uzoraka

Kao veoma bitan zadatak u današnjoj IT industriji postavlja se prepoznavanje zvuka od strane računara. Veliki broj komercijalnih programa na ovom polju se kreće u pravcu prepoznavanja ljudskog glasa (engl. voice recognition). Ovakav softver bi omogućio lako izdavanje naredbi glasom bilo kom uredjaju i na taj način doneo uštedu vremena izbacivši tastaturu, miša, elektronsku olovku i ostale alate kojima se obraćamo elektronskim uredjajima.

Sa druge strane razvija se softver za prepoznavanje muzičkih kompozicija. Svakom se desilo da se nađe u klubu, na koncertu, sluša pesmu na radiju i očajnički pokušava da se seti i prepozna pesmu koju sluša. Zahvaljujući brzom razvoju mobilnih telefona i aplikacijama na njima, ovaj uređaj je postao

veoma pogodan za snimanje muzike, a samim tim i prepoznavanje muzičkih uzoraka. Trenutno su na tržištu najpopularniji konkurenti Shazam i SoundHound [7,8]. Oba programa su trenutno dostupna u besplatnim verzijama za platforme iOS, Android i Windows phone. Ove aplikacije snimaju zvuk putem mikrofona, upoređuju ga sa trenutnim zapisima iz baze podataka i vraćaju pesmu koja je najbliža zadatoj. U najvećem broju slučajeva potrebno je snimiti nekih 20 sekundi pesme da bi došlo do pogotka. Najbolji rezultat postiže se snimanjem studijske verzije pesme u tihoj prostoriji bez spoljnih šumova. Ipak, softver je pokazao zavidne rezultate i u prepoznavanju pesama koje su snimljene u mnogo težim uslovima (na koncertima, u klubovima i bučnim prostorijama).

Shazam Entertainment, Ltd je Američka kompanija, koja je nastala 2000. godine sa idejom da što više približi ljudima različite žanrove muzike. Programeri iz ove kompanije govore da je njihov algoritam otporan na spoljašnje šumove i izobličenja i da poseduju bazu od preko 2 miliona pesama. Svaka od ovih pesama okarakterisana je jedinstvenim digitalnim potpisom na osnovu koga se izvršava pretraga. U ovom radu akcenat će biti na implementaciji algoritma za prepoznavanje muzičkog uzorka koji se direktno oslanja na algoritam koji trenutno koristi Shazam.

Glava 2

Prepoznavanje muzičkih uzoraka

Ovo poglavlje navodi ključne delove programskog koda koji je zapisan u obliku pseudo koda. Sledi lista korisničkih klasa i metoda koje su korišćene:

- AudioFormat - odredjuje format audio zapisa i sastoji se parametara: `frekvencijaUzorka`, `velicinaUzorkaUBitovima`, `brojKanala`, `oznaceno`, `bigEndian`. Metoda `vratiFormat()` vraća format koji će se koristiti za programsku realizaciju.
- LinijaPodataka - stream(deo JAVA sound biblioteke) koji se otvara za snimanje zvuka u datom formatu.
- AudioSystem - klasa JAVA sound biblioteke koja ima pristup svim linijama i audio mikserima(alati za kombinovanje zvukova snimljenih na različitim kanalima, dodavanje zvučnih efekata) unutar sistema. Njena funkcionalnost proširena je Tritonus bibliotekom za rad sa MP3 formatom [9].
- IzlazniStrim - Tzv. output stream iz java i/o biblioteke. Metoda prebaciUByteArray() transformiše izlaz u niz bajtova.
- ByteArrayIzlazniStrim - Nasledjuje gore navedeni izlazni strim i obezbedjuje da izlaz bude niz bajtova
- KompleksnaMatrica - Dvodimenzioni niz kompleksnih brojeva
- KompleksniNiz - Jednodimenzioni niz kompleksnih brojeva
- FFT - Klasa za primenu brze Furijeove transformacije. Metoda fft uzima kompleksni broj kao argument i vraća niz.

- TrenutakPesme - jednoznačno odredjuje trenutak(vreme) odredjene pesme(sa jedinstvenim Id-om)

2.1 Programsko snimanje zvuka

S obzirom na to da se za snimanje najčešće koristi format wav, koji je pritom najfleksibilniji za samu obradu, izabran je i za potrebe ovog rada. Pri snimanju, zbog uštede prostora na hard disku kvalitet zvuka će biti smanjen na 8 bita umesto 16 po uzorku, jednokanalni zvuk.

U programskom jeziku JAVA postoji biblioteka "Java Sound"koja poseduje sve što je potrebno za rad sa muzičkim uzorcima. Ova biblioteka kontroliše ulaz i izlaz audio podataka. Poseduje mehanizme za instalaciju, pristup i kontrolu audio mikseru, MIDI uredjajima (instrumentima povezanim na zvučnu karticu) kao i konvertore za razne formate zvuka. Pronašla je veliku primenu u konferencijskom softveru, telefoniji, muzičkim plejerima i igrama.

```
 AudioFormat vratiFormat() {
    float frekvencijaUzorka = 44100;
    int velicinaUzorkaUBitovima = 8;
    int brojKanala = 1; //jednokanalni zvuk
    boolean oznaceno = true;
    boolean bigEndian = true;
    return AudioFormat(frekvencijaUzorka, velicinaUzorkaUBitovima,
    brojKanala, oznaceno, bigEndian);
}

AudioFormat format = vratiFormat();
LinijaPodataka linija = AudioSystem.vratiLiniju(format);
linija.start();
```

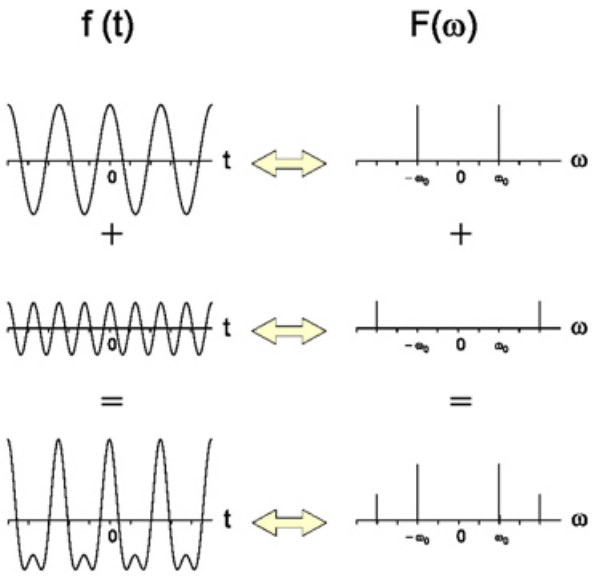
može da se počne sa snimanjem.

```
IzlazniStrim izlaz = ByteArrayIzlazniStrim();
tekuci = true;
try {
    while (tekuci) {
        int brojBajtova = linija.procitaj(bafer, 0, duzinaBafera);
        if (brojBajtova > 0) {
            izlaz.ispisi(bafer, 0, brojBajtova);
        }
```

```
    }
    izlaz.zatvori();
} catch (Izuzetak e) {
    Ispisi("Problem sa Ulazom/Izlazom");
}
```

2.2 Transformisanje zvučnog signala u frekventni domen primenom FFT

Kao što je spomenuto u 1.2, za prebacivanje signala u frekventni domen koristi se FFT. Slika 3 prikazuje grafike funkcija u zavisnosti od vremena i frekvencije.



Slika 2.1: Primeri signala u vremenskom i frekventnom domenu

FFT se primenjuje nad grupom uzoraka (obično ih nazivamo okvirom). Okvir transformišemo u kompleksni vektor, odnosno realni i imaginarni deo, koji predstavljaju amplitudu i fazu. Kao krajnji rezultat dobijeno je onoliko frekvencija koliko je bilo uzoraka u okviru.

Za potrebe Furijeove transformacije napisana je korisnička klasa FFT koja poseduje metodu za pozivanje Furijeove transformacije. Treba napomenuti da je za veličinu okvira uzeto 4096 bajtova.

```
byte audio[] = izlaz.prebaciUByteArray();
```

```
int ukupnaVelicina = Velicina(audio);
int velicinaPoCanku = ukupnaVelicina/VelicinaCanka;
KompleksnaMatrica[][] rezultat = KompleksnaMatrica[velicinaPoCanku][];
for(int j = 0;i < velicinaPoCanku; j++) {
    KompleksniNiz[] kompleksniNiz = KompleksniNiz[velicinaPoCanku];
    for(int i = 0;i < velicinaPoCanku;i++) {
     KompleksniNiz[i] = KompleksniBroj(audio[(j*velicinaPoCanku)+i], 0);
    }
    rezultat[j] = FFT.fft(KompleksniBroj);
}
```

2.3 Izdvajanje značajnih frekvencija muzičkog uzorka

Posle izračunavanja FFT-a, sledeći korak bi bio izdvajanje ključnih delova pesme u sastavljanje heš taga od tih delova. Naravno, na kraju je potrebno potražiti u bazi entitet sa istim heš tagom. Ključan deo programa jeste formiranje odgovarajuće heš funkcije koja jednoznačno preslikava jednu pesmu u niz heš tagova.

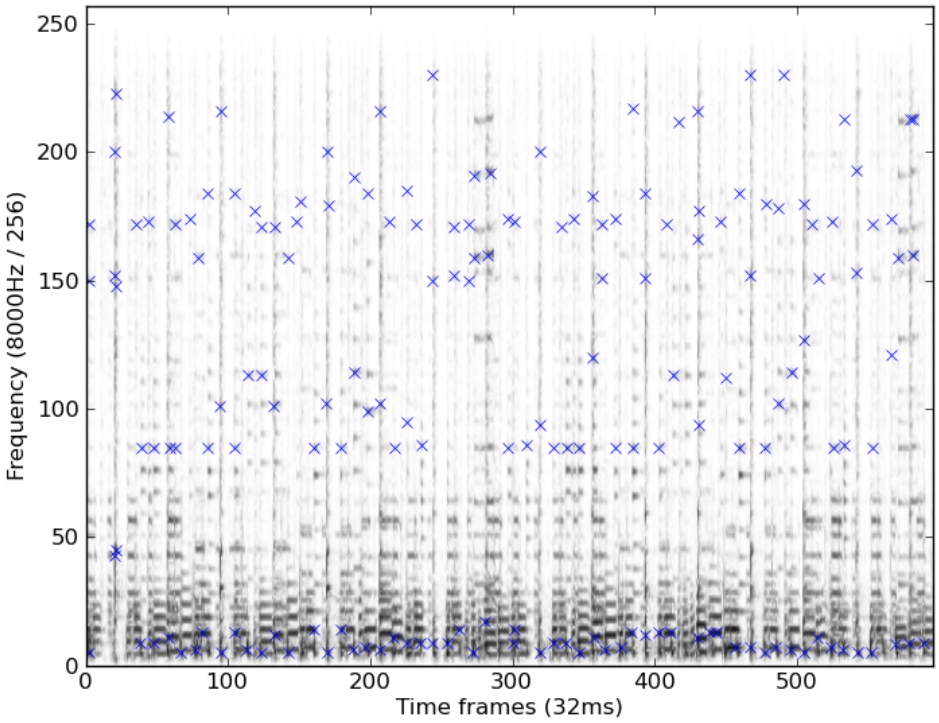
Jedan od problema koji se javlja jeste kako izabrati frekvencije koje su karakteristične za određeni zvuk. Potrebno je zadržati vremensku dimenziju, translacionu invarijantnost i uređenost skupa [5]. Prvi uslov podrazumeva informaciju u kom trenutku u odnosu na početni trenutak se pojavila neka frekvencija. Translaciona invarijantnost znači da pri snimanju ulaznog zvuka ne moramo voditi računa o tome da li pesma kreće od početka, već da je na osnovu "offsetinga"(pomeraja) moguće prepoznati tačnu poziciju snimljenog zvuka u zapisanoj heš tag vrednosti(dovoljno je na primer pustiti samo refren pesme). Skup mora biti uređen(zvučni signali moraju posedovati jednoznačnu odredjenost) zato što se u bazi podataka mogu naći milioni pesama sa sličnim frekvencijama u sličnim vremenskim trenucima. Ako se ne izvrši prečišćavanje nekih frekvencija, može doći do velikog broja rezultata među kojima je teško izabrati onaj pravi.

Pretpostavljeno je da se snimanje neće uvek odvijati u idealnim uslovima, tako da može doći do pojave šuma u snimku. Pošto uređaji za snimanje nisu savršeni, još jedan problem je izobličenje (rastezanje zvuka). Ove dve prepreke se mogu prevazići ako se za sastavljanje heš taga uzmu najviše frekvencije u određenim intervalima [6].

2.4 Heširanje muzičkog uzorka

Ljudsko uho je sposobno da registruje frekvencije izmedju 20 i 20 000 hz. Najniži ton koji se može odsvirati na klaviru je C1 (32.70hz) dok je najviši C8 (4186.01 Hz. U obzir je potrebno uzeti i izvor emitovanja zvuka (zvučnik, pojačalo). Za potrebu ovog rada pretpostavićemo da se sve dešava na intervalu 30 - 300hz. Potrebno je izvršiti podelu ovog domena na intervale tako da se dobije što manji raspon u "važnim intervalima", odnosno intervalima u kojima ćemo tražiti pikove. Ako se uzme da su niži tonovi (npr. bas linija, koja predstavlja tonove proizvedene uz pomoć bas gitare) bitniji, s obzirom na to da prate glavnu temu pesme, izabrana su prva tri intervala (30hz -40hz), (40hz - 80hz) i (80hz - 120hz). Uzima se još jedan interval za srednje tonove (120hz - 180hz) i jedan za visoke (180hz - 300hz) [5]. Ovi intervali su dobijeni detaljnim ispitanjima od strane inženjera kompanije Shazam. Odabiranjem najviših od frekvencija iznad praga (pikova) u svakom od ovih intervala dobijamo jednu heš tag vrednost. Na slici 4 prikazan je zvučni signal sa obeleženim pikovima. X osa predstavlja vremensku dimenziju gde je jedinica okvir veličine 32ms, dok Y osa predstavlja frekvencije izražene u hercima.

Nakon snimanja dela pesme 'Stairway to heaven', pa zatim primenom FFT algoritma i izdvjanjem najviših frekvencija iz datih intervala dobija se sledeći niz heš tagova:
33 56 99 121 195
30 41 84 146 199
33 51 99 133 183
33 47 94 137 193
32 41 106 161 191
33 76 95 123 185
40 68 110 134 232
30 62 88 125 194
34 57 83 121 182
34 42 89 123 182
33 56 99 121 195
30 41 84 146 199
33 51 99 133 183
33 47 94 137 193
32 41 106 161 191
33 76 95 123 185



Slika 2.2: Snimljeni uzorak u frekventnom domenu sa ucrtanim pikovima

Primer jednog heš taga: 33 56 99 121 195. Sledi deo koda koji određuje pikove u pet navedenih intervala.

```
for (interval in listaIntervala)
  for (f = donjiLimit; f < gornjiLimit; f++)
     zvuk = rezultat[f];
        index = vratiIndex(zvuk);
        if (zvuk > maxZvuk){
         maxZvuk = zvuk;
            hesTagovi[index] = zvuk;
            }
```

Prva petlja prolazi kroz listu intervala (prethodno zadati intervali u kojima tražimo pikove), dok druga prolazi kroz pojedinacni interval tražeći u njemu maksimalnu vrednost frekvencije(pik). Niz `rezultat` je niz frekvencija dobijen prethodno primenom FFT, dok funkcija `vratiIndex` vraća broj intervala u kome se nalazi zvuk (od 0 do 4).

2.5 Upoređivanje heš vrednosti, pronalaženje u bazi podataka

Pretpostavlja se da postoji baza podataka sačinjena od osam miliona pesama, i da se pri snimanju pesama koristi opisani princip(u samoj programskoj realizaciji se koristi baza podataka od 20 pesama ali je u praktičnom slučaju taj broj puno veći). Jedan entitet u bazi će biti upravo heš tag sačinjen od 12 ili 13 brojeva(maksimumi svakog od intervala spojeni u jedan heš tag). Da li će se u jednom heš tagu naći 12 ili 13 brojeva zavisi od maksimuma intervala [80, 120] koji može biti dvocifren ili trocifren. Trenutak je da se uvede i takozvani "faktor šuma", gde šum predstavlja neželjenu smetnju koja se preklapa sa korisnim signalom. Za ovu demonstraciju je uzet ostatak pri deljenju sa 2, što znači da se za brojeve u heš tagu uzimaju samo parni brojevi (Shazam ne otkriva tačno koji faktor šuma se koristi). Tada formiranje heš taga igleda ovako:

```
FAKTOR_SUMA = 2

formirajHes(){
    String p[] = pikovi.splitujPoIntervalima();
    pikPrvogIntervala = p[0]; //(30hz -40hz)
    pikDrugogIntervala = p[1]; //(40hz - 80hz)
    pikTrecegIntervala = p[2]; // (80hz - 120hz)
```

```
    pikCetvrtogIntervala = p[3]; // (120hz - 180hz)
    pikPetogIntervala = p[4]; // (180hz - 300hz)
    hesTag = (pikPetogIntervala-(pikPetogIntervala mod FAKTOR_SUMA))
    * 10000000000 + (pikCetvrtogIntervala-(pikCetvrtogIntervala mod
    FAKTOR_SUMA))* 10000000 + (pikTrecegIntervala-(pikTrecegIntervala
    mod FAKTOR_SUMA)) *10000 + (pikDrugogIntervala-
    (pikDrugogIntervala mod FAKTOR_SUMA)*100+ pikPrvogIntervala -
    (pikPrvogIntervala mod FAKTOR_SUMA));
}
```

Dakle najpre se pikovi izdvajaju po intervalima i ubacuju u niz `p`(uzet za privremeno smeštanje pikova) pri čemu je niz pikova dobijen prethodnim izračunavanjem
Zatim se formira heš tag uz korekciju dobijenu ostatkom pri deljenju sa faktorom šuma. Potrebno je kreirati strukturu podataka `Map(Long, List(TrenutakPesme))`. `TrenutakPesme` predstavlja uredjeni par IdPesme, vreme u pesmi.

```
class TrenutakPesme {
    private int vreme;
    private int IdPesme;

    public TrenutakPesme(int IdPesme, int vreme) {
        this.IdPesme = IdPesme;
        this.vreme = vreme;
    }

    public int vratiVreme() {
        return vreme;
    }
    public int vratiIdPesme() {
        return IdPesme;
    }
}
```

Dakle ključ mape predstavlja sam heš tag dok je vrednost lista instanci klase `TrenutakPesme`. Sledeće pitanje koje se postavlja je kako posle pokretanja programa izabrati pravu pesmu iz baze podataka(fajla) gde svaka pesma zauzima oko 10 kb i zapisana je kao niz bajtova u fajlu. Jedan od kriterijuma bi bio broj pogođenih heš tagova u datom vremenskom rasporedu. S obzirom na to da postoji preko 1000 heš tagova po jednoj pesmi postoji velika verovatnoća da se heš tagovi dva potpuno različita muzička uzorka poklope.

Definitivno je potrebno uzeti u obzir i vremensku dimenziju uzorka. Pravi pogodak bi bio onaj koji zadovoljava sledeći uslov:
Ako je snimljenZvuk niz heš tagova koji predstavlja uzorak koji se trenutno ispituje i dbZvuk niz heš tagova koji predstavlja uzorak koji se nalazi u bazi (fajlu), potrebno je odrediti takozvani 'offseting' zapisa odnosno vremenske pozicije heš tagova . Pogodak je ostvaren ako važi da su pronadjena dva podudarna heš taga na jednakim vremenskim razmacima.
snimljenZvuk[i1] = dbZvuk[j1] i snimljenZvuk[i2] = dbZvuk[j2] i i2 - i1 = j2 - j1

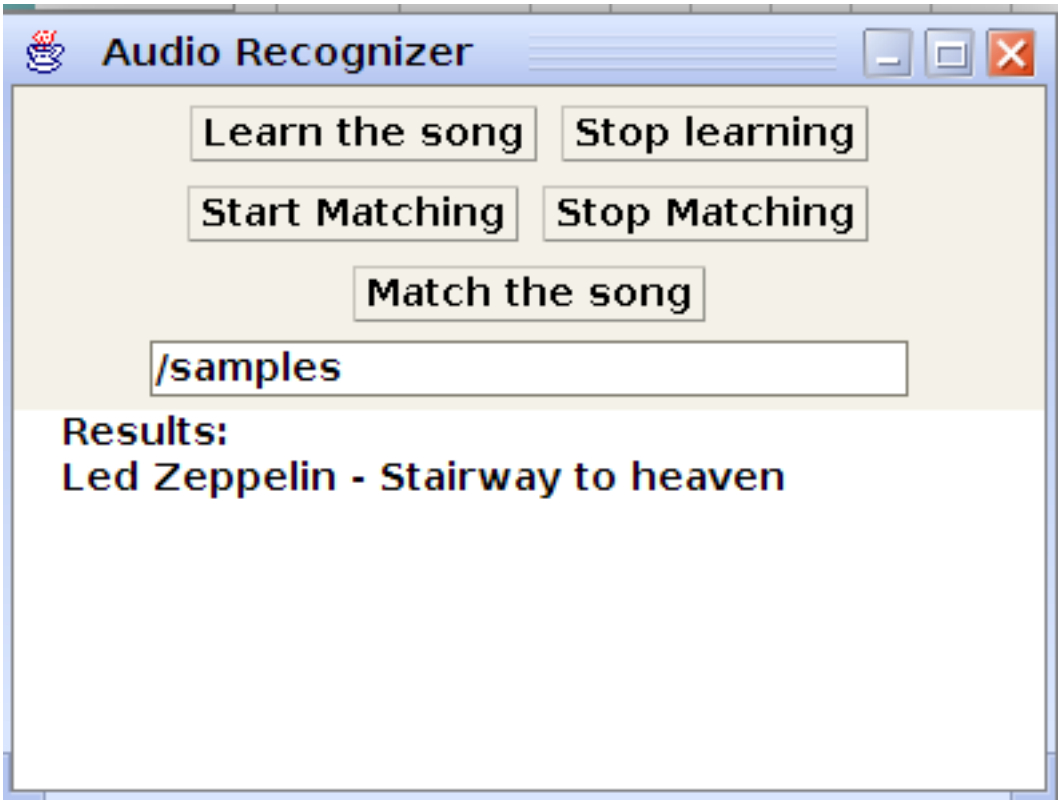
Glava 3

Programska realizacija i pregled dobijenih rezultata

Za programsku realizaciju korisćen je programski jezik Java. Jedan od razloga jeste dostupnost velike količine besplatnih biblioteka, između ostalih i "java sound"biblioteke koja je korišćena za snimanje materijala. Sam algoritam nije preterano komplikovan, a rešenje je za sada više dokaz koncepta nego kompletirana aplikacija spremna za prezentaciju pred korisnicima.

3.1 Opis programske realizacije

Sam zvučni zapis se beleži u fajl. Ranije pomenuta biblioteka 'Java sound' se koristi u ovom slučaju samo za snimanje. Treba pomenuti njenu širu primenu u reprodukciji muzičkih fajlova, audio miksovanju kao i kontroli MIDI uredjajima (npr. gitara povezana na računar uz pomoć zvučne kartice). Program ima opciju učitavanja i obrade neke melodije sa određene URL adrese ili iz iz samog mp3 fajla na disku. Za čitanje mp3 fajla se koristi eksterna biblioteka "tritonus". Pošto se završi proces učenja(punjenja baze melodijama), može se snimiti preko mikrofona zadati uzorak, uz prethodno opisano transformisanje signala u frekventni domen, određivanje ključnih tačaka i formiranje heš tag vrednosti. Za potrebe Brze Furijeove transformacije napisana je klasa FFT.java koja sadrži metod za konverziju u frekvencijski domen. Heš tag vrednosti se porede sa prethodno naučenim vrednostima i u odnosu na "offset"(poziciju u pesmi) kalkulišu broj tzv. mečeva(slaganja u pesmi). Drugim rečima meč je pronadjena heš tag vrednost u zadatom vremenskom rasporedu. Pesma sa maksimalnim brojem mečeva je prvi kandidat za prepoznatu pesmu, ali je u rezultatu moguće prikazati i moguće alternative za slučaj da algoritam nije najbolje obavio svoj posao. Trenutna verzija radi



Slika 3.1: Korisnički interfejs aplikacije

sa zvukom snimljenim u jednokanalnom formatu, zbog ograničenja javine biblioteke za obradu mp3 fajla.

3.2 Prikaz eksperimenata sa razvijenim programom

Slika 5 prikazuje izgled osnovnog prozora aplikacije. Korisnički interfejs se sastoji iz pet dugmeta, jednog tesktualnog input polja i jedne tekstualne površine. Za samu implementaciju korišćena je biblioteka swing. Aplikacija uči o zvučnom zapisu nakon klika na dugme 'Learn the song' a završava sa učenjem klikom na 'Stop Learning'. Pri tom je potrebno navesti putanju do mp3 fajla ili URL na kome se nalazi zvučni zapis. Format mp3 fajla mora biti u jednokanalnom formatu sa veličinom uzorka od 8 bitova i učestanošću odmeravanja od 44100 hz. Nakon unosa uzoraka, proces traženja odgovarajuće melodije u bazi se započinje klikom na dugme 'Start Matching' kojim kreće snimanje preko mikrofona a završava klikom 'Stop Matching'. Konačno

klikom na 'Match the song' pojavljuje se rezultat.

Glava 4

Zaključak

Razvoj mobilnih tehnologija pruža mogućnost korišćenja aplikacija za prepoznavanje zvuka na bilo kom mestu u bilo koje vreme. Današnje aplikacije za android i iPhone su otišle i korak dalje, pa je moguće dati naredbu telefonu sopstvenim glasom. U budućnosti se definitivno očekuju računari, pa čak i kućni uređaji poput veš mašine, televizora ili bojlera, koji će naredbe primati uz pomoć ljudskog glasa.

Što se same muzičke umetnosti tiče, stiče se utisak da pomalo zapada u kreativnu krizu. Novi pravci poput elektronske i tehno muzike koriste tehnologiju umesto živih instrumenata. Možda je tehnologija jednostavno suvišna u samom procesu kreiranja muzike, ali je definitivno treba iskoristiti na najbolji mogući način za njenu obradu. Postavlja se pitanje kako su Mocart i Betoven dobili inspiraciju za "Malu noćnu muziku"i "Mesečevu sonatu"i odakle nadahnuće: Džonu Lenonu, Džimiju Pejdžu, Džimiju Hendriksu i ostalim zvezdama rok muzike. Muzičari tvrde da njihove kompozicije nastaju u posebnim trenucima telesne opuštenosti kada se njihovo trenutno raspoloženje i stanje duše ispoljava kroz tonove. Svaki ton i svaki akord jeste produkt prethodno odslušanih tonova i akorda i zato danas muzičari otvoreno pričaju o uticaju pod kojim su stvarali neko delo. Da li je moguće napraviti softver koji bi potencijalno mogao da pronađe sličnost izmedju dve kompozicije? Da li bi taj softver dao odgovor na pitanje ko su "Adam i Eva"muzike i pod čijim su uticajem bili prvi nama poznati kompozitori ovog sveta?

Dalji pravci rada se zasnivaju na detaljnijoj analizi heš tag vrednosti. Formiranjem određenih koeficijenata rastezanja bilo bi moguće prepoznati pesmu koja nije snimljena u istom tempu kao original. Trenutna verzija algoritma se pri traženju melodije u bazi u potpunosti oslanja na "offset"pesme, odnosno relativni trenutak u kome se dogodila neka frekvencija, ali skaliranjem "offseta"određenim koeficijentom moguće je učiniti algoritam fleksbil-

nijim. Jedna od ideja koja se nameće je i klasifikacija muzike po žanrovima na osnovu heš tagova. Prepoznavanje sličnih pesama bi svrstalo odredjene numere u klasičnu, rok, pop, džez ili folk muziku. Jedna od primena ovog rada mogla bi se naći i u prepoznavanju filmova na osnovu zvuka. Ipak za tako nešto potrebno je izdvojiti ogroman prostor za smeštanje digitalnih potpisa celih filmova ili samo filmskih sekvenci. Sa druge strane eksponencijalni razvitak tehnologije doneće nova uzbudjenja i ono što danas izleda nemoguće postaće deo svakodnevice u budućnosti. Već je moguće izdavati glasovne komande mobilnim telefonima, kućnim uredjajima a nije daleko ni trenutak kada će se programirati u sličnom maniru.

Bibliografija

[1] M. Pilhofer, Music Theory for Dummies, For Dummies, 2007

[2] T. Jelavić, Zvuk, sluh, arhitektonska akustika, Školska knjiga, 1978

[3] D. Radunović, Numeričke metode, Matematički fakultet, 2004

[4] E. Grosvenor, A. G. Bell: The Life and Times of the Man Who Invented the Telephone, Harry N Abrams, 1997

[5] Avery Li-Chun Wang, An Industrial-Strength Audio Search Algorithm `http://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf`

[6] J. Haitsma, Ton Kalker, A Highly Robust Audio Fingerprinting System `http://www.nhchau.com/files/AudioFingerprint-02-FP04-2.pdf`

[7] SH SoundHound - INstant music search and discovery, `http://www.soundhound.com/`

[8] Shazam, http://www.shazam.com/

[9] Biblioteka Tritonus, http://www.tritonus.org/

[10] M. Park, Compare Music File Formats, University of Texas – Austin, 2006

[11] `http://laplacian.wordpress.com/2009/01/10/how-shazam-works/`

[12] J. Hogson, Understanding Records, Bloomsbury Academic, 2010