UNIVERZITET SINGIDUNUM

Fakultet za informatiku i računarstvo

PRETRAŽIVANJE MUZIČKIH MATERIJALA PO SADRŽAJU

− diplomski rad −

Mentor:

Prof. dr Vladislav Miškovic

Kandidat:

Dejan Čugalj

Br. indeksa: 2010/200802

Beograd, 2019.

Sadržaj

1. Uvod ..... - 1 -

1.1 Problem pretraživanja po sadržaju ..... - 1 -

1.2 Struktura rada ..... - 3 -

2. Teorijske osnove digitalizacije zvuka ..... - 4 -

2.1 Svojstva zvuka i fizičke karakteristike ..... - 5 -

2.1.1 Fizičke veličine zvuka (*subjektivne karakteristike*) ..... - 6 -

2.1.2 Fizičke karakteristike zvuka (*objektivne karakteristike*) ..... - 6 -

2.2 Digitalizacija ..... - 8 -

2.2.1 Odmeravanje – diskretizacija po vremenu ..... - 9 -

2.2.2 Kvantizacija – diskretizacija po amplitudi ..... - 11 -

2.2.3 Kodovanje – Linearna impulsna kodna modulacija ..... - 14 -

3. Priprema digitalnog audio signala za pretraživanje ..... - 15 -

3.1 Spektrogramski otisak zvuka ..... - 15 -

3.2 „Otisak prsta“ audio signala ..... - 16 -

3.2.1 Frekvencijska predstava digitalnog signala ..... - 18 -

3.2.2 Prozorske funkcije ..... - 21 -

3.3 Ekstrakcija jedinstvenih tačaka iz audio signala ..... - 23 -

3.4 Spektrogramsko filtriranje i kreiranje heš vrednosti ..... - 24 -

3.4.1 Heširanje vrednosti ..... - 24 -

3.5 Podudaranje ..... - 25 -

4. Praktična realizacija aplikacije za pretraživanje muzičkih materijala ... - 26 -

4.1 Digitalno učitavanje audio sadržaja ..... - 27 -

4.2 Učitavanje audio datoteka za indeksiranje i pretragu ..... - 28 -

4.2.1 Kreiranje FFT spektrograma ..... - 29 -

4.2.2 Ekstrakcija važnih informacija iz audio signala ..... - 30 -

4.2.3 Memorisanje ekstraktovanih informacija iz audio signal ..... - 32 -

4.2.4 Pretraga audio materijala po sadžaju ..... - 33 -

5. Zaključna razmatranja ..... - 34 -

Literatura ..... - 35 -

Dodatak I

Listing 1. I

Listing 2. II

Listing 3. III

Listing 4. III

Listing 5. IV

Listing 6. IV

1. Uvod

Primarni zadatak ovog diplomskog rada je da sa praktične strane predstavi primer implementacije pretrage muzičkih materijala po sadržaju koji je napisan u radu [1] i razvije eksperimentalna aplikacija u programskom jeziku Java, koje je primenjiva u oblasti pretrage audio materijala po sadržaju.

Problemi koji se javljaju prilikom pretrage muzičkih materijala su veoma zanimljivi i daju jedinstveni primer kako se kreativnim pristupom može doći do relativno jednostavnog rešenja komplikovanog zadatka. Kreativnim pristupom problemu i ekstrakciji samo „bitnih", ključnih informacija iz podataka, naprednim metodama „heš" funkcija specifičnih za audio materijale i kreiranje audio otiska, podstiču razmišljanje ka rešavanju skoro svakog problema na koji se može naići ove prirode.

1.1 Problem pretraživanja po sadržaju

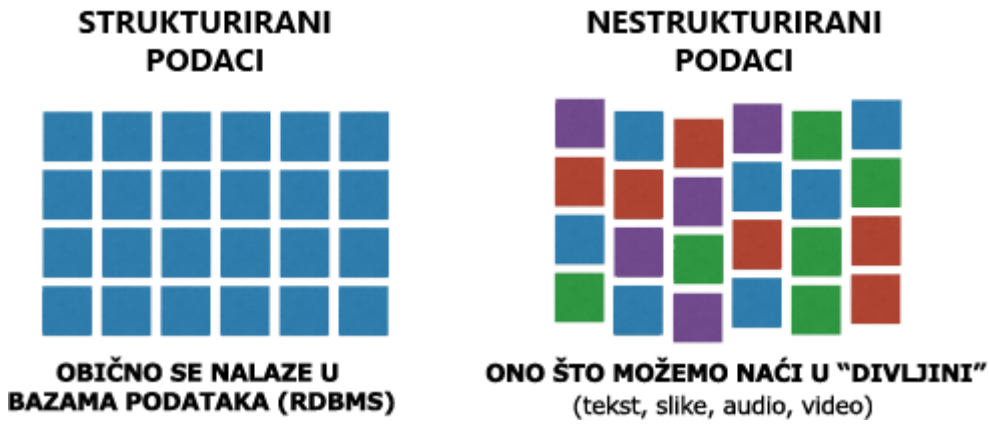
Digitalna pretraga predstavlja organizovanu potragu po uređenim ili neuređenim strukturama podataka u kolekciji dokumenata kao što su Internet stranice, računarske datoteke, video sadržaji, muzički sadržaji itd. Dobijeni rezultat pretrage je podatak, koji u većini slučajeva ne znači mnogo, ali onog momenta kada se upotrebi dobija status informacije. Informacije su upravo oni gradivni elementi sa kojima se stiže do višeg stepena u hijerarhijskoj „*PIZM*" piramidi, a to je znanje [2].

Slika 1. Hijerarhisjki model „PIZM"

Dobijanje informacija iz strukturiranih podataka je u velikoj većini slučajeva samo „računarski posao" i ukoliko se računaru daju određena i tačna uputstva, pod pretpostavkom da se raspolaže „razumnom" računarskom snagom, relativno je lako doći do njih. Drugim rečima, informacija se dobija tako što se analiziraju veze i odnosi između podataka [2]. Problemi se javljaju u slučajevima kada su podaci nestrukturirani i po pravilu su oni jako veliki, pa te veze i odnosi nekada nisu odmah „vidljivi-uočljivi", nego je potreban kreativniji pristup dolaženja do istih. Znanje koje se dobija delovanjem nad informacijama je u današnjem društvu je toliko značajno, da informaciju možemo posmatrati kao dominantan resurs u svim oblastima društvenog i naučnog života, ali isto tako i jedan od ciljeva današnjeg informacionog društva koji predstavlja još jedan stepenik ka ultimativnom dostizanju vrha hijerarhijskog „*PIZM*" piramidalnog modela, Slika 1, a tek onda može da se kaže da je podatak zaista koristan i da u njemu postoji upotrebljiva informacija, iz koje se nešto saznaje.

Strukturirane podatke koji se nalaze u bazama podataka je moguće pretraživati tradicionalnim metodama, kao što je postavljanje upita (eng. *query*), međutim kako se danas podaci generišu neverovatnom brzinom i u još većem obimu, pa su to još i u velikoj većini nestrukturirani podaci, dolazi se do zaključka da je izvlačenje informacija iz ovakve vrste podataka tradicionalnim metodama skoro neizvodljivo.

Strukturirani podaci su oni podaci čiji je model i format unapred definisan. Kreirani su u formatu tako da bi se mogli lako organizovati, analizirati i pretraživati. Ovakva vrsta podataka se obično nalazi u relacionim bazama podataka (*RDBMS*), razvrstani po tabelama i poljima. Imena i tip polja relacione baze, određuju vrstu podatka koji se smešta u tabele. Ovakav unapred definisan format je veoma lako pretraživati pomoću ljudski generisanih upita ili pomoću algoritama koji koriste tip podatka i imena polja kao parametre za pretragu.



Slika 2. Strukturirani i nestrukturirani podaci

Nasuprot strukturiranim podacima su nestrukturirani podaci. Nestrukturirani podaci se često viđaju u tekstualnim zapisima i multimedijalnim sadržajima, Slika 2. Primeri ovakvih podataka su:

- Elektronska pošta
- Tekstualno procesuirani dokumenti (eng. *Word Processing Documents*)
- Video zapisi
- Fotografije
- Muzičke datoteke
- Prezentacije
- Internet stranice

Pretpostavlja se da je ovakav vid podataka zastupljen čak u 70%-80% od ukupno svih podataka koji su trenutno generisani ili se generišu. Poznato je da svaki nestrukturirani digitalni zapis u vidu slike, videa, muzike, e-knjige itd. sadrži neke od metapodatka (eng. *metadata*). Metapodaci su kratki opisni atributi koji dodatno opisuju datoteku i obično su to kratki podaci kao što su: *ime autora*, *vreme kreiranja*, *veličina datoteke* itd.

Do metapodataka je programerskim putem prilično lako doći i oni na prvi pogled mogu da daju rešenje problema istraživanja ovog rada, tako što bi jednostavnim izvlačenjem metapodataka iz datoteka i komparacijom sa predefinisanim skupom iz baze podataka, se došlo do rezultata pretrage.

Međutim, problemi koji se javljaju prilikom implementacije ove metode, izražavaju se u tome da nikako nije zagarantovana tačnost tih kratkih opisnih atributa, takođe su u većini slučajeva i nepotpuni. Metapodaci su integrisani u samu datoteku, pa pretraga nije moguća po sadržaju, nego samo po opisnim atributima, tj. metapodacima.

1.2 Struktura rada

Diplomski rad sadrži pet poglavlja, literaturu i listinge koda u dodatku. Akcenat je stavljen na muzičke datoteke.

Drugo poglavlje se fokusira na teorijske osnove digitalizacije zvuka. Predstavljene su fizičke karakteristike i opšta svojstva zvuka. Takođe je predstavljen proces digitalizacije kontinualnih talasa kroz prizmu audio materijala.

Treće poglavlje predočava, šta je to jedinstveno u audio signalima i kako ih pripremiti za pretraživanje. Ta jedinstvenost će kasnije i omogućiti implementaciju algoritma potrebnog za ovaj rad. Razmatran je spektrogramski otisak zvuka, kao nosilac jedinstvenih informacija signala. Matematička predstava u kreiranju spektrograma je dala uvid u frekvencijske signalne predstave, koji su korišćeni za kreiranje „otiska prsta“ posmatranog audio talasa.

U četvrtom poglavlju je predstavljena eksperimentalna aplikacija kao i njen programerski deo u obliku Java koda. Programerski je prikazano kako napisati FFT funkciju, ekstraktovati važne informacije kao i memorisanje heš vrednosti. Sam kraj četvrtog poglavlja je rezervisan za pretragu audio materijala po sadržaju.

2. Teorijske osnove digitalizacije zvuka

Platonov učenik, Aristotel[1] je još u doba pre nove ere smatrao da je kretanje svetlosti i zvuka slično talasima na morskoj površini. Takođe je predvideo da je prostiranje zvuka nemoguće u sredini u kojoj ne postoji medijum za prenos, kao što su vazduh ili voda (elastična sredina). Tek u XVII veku je uspešno izveden eksperiment gde je „stvoren" vakum ispod staklenog zvona u koji je ubačen mehanički sat. Zvuk kucanja sata ispod vakumskog staklenog zvona je postao beščujan, što je ujedno i potvrdilo Aristotelovu tvrdnju da je prenošenje zvuka bez nekog medijuma za prenos, zaista nemoguć.

Eksperimentalni dokaz Aristotelove tvrdnje govori da je od krucijalnog značaja izvora i prenosa svakog zvuka i zvučnog talasa postojanje medijuma za prenos tj. elastične sredine, gde delovi sredine (molekuli), izvedeni iz svog ravnotežnog položaja (vibracija-oscilovanje*)* prenose zvučni talas. Talas kao mehaničko pobuđivanje elastične sredine u kome se prenosi, može da se prostire kao:

- Longitudinalni talas
- Transverzalni talasi
- Fleksioni talasi
- Površinski talasi
- Ekstenzioni talasi

Kako se elastične sredine mogu podeliti na čvrsta tela i fluide, a pod fluidima se podrazumevaju gasovi i tečnosti, jedino longitudinalni talas[2] može da se prostire kroz sva tri stanja medijuma za prenos (čvrsto-tečno-gasovito). Longitudinalni talas kao nosilac zvučnog signala, predstavlja kanal za prenos „kriptovanih" podataka u obliku zvuka, a da bi se „dešifrovali" zahteva neku vrstu dekripcionog sistema. Upravo taj dekripcioni sistem je ljudski rod prirodnom evolucijom dobio u obliku sluha. Čovečiji slušni organi su primer „prijemnika" koji može da registruje vibracije talasa, koji se prenose gasovitom elastičnom sredinom, i dekodira ih u zvučni podatak.

Svi izvorni zvučni podaci koji dolaze do slušnih organa su analogni signali, kontinuirani talasi kao predstava maksimalne prosečne količine informacija koje mogu da se prenesu od izvora (zvuk), do ulaza (sluh). Takvi signali nemaju nikakav gubitak u kvalitetu prenosa informacija, ukoliko se zanemari šum koji se javlja prilikom prolaska kroz medijum za prenos. Obzirom da je ovo samo analogni signal, kontinualan talas, čuvanje ovakvog signala na nekim od računarsko memorijskim medijumima predstavlja isto što i čuvati beskonačnu količinu podataka, što trenutno ipak nije izvodljivo-praktično. Istovremeno, analogni signal je neupotrebljiv za osnovni element računarske tehnologije mikroprocesor, koji „ne razume" analogne vrednosti i kao takve nikako nije u mogućnosti da pretvori podatak u informaciju. Nemogućnost procesora da pretvori podatak u informaciju uskraćuje bilo kakvu manipulaciju sa istim, pa je potrebno da se takvi signali konvertuju u računarsko razumljiv domen, koji se naziva „digitalni audio".

[1] Aristotel (grč. Ἀριστοτέλης) (384. pne.-322. pne.), naučnik i filozof Antičke Grčke.
[2] Talas koji se širi u istom pravcu u kojem se kreću molekuli medijuma pri oscilovanju.

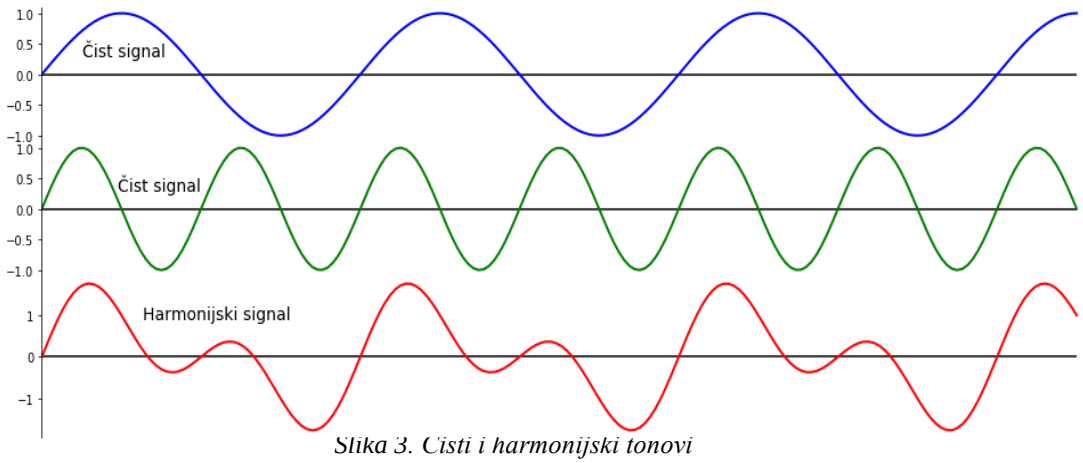
Elektronska prezentacija zvuka je omogućila da se analogni signal pretvori u brojnu vrednost, tj. da se zvučni talas binarno prezentuje. Proces koji je još poznat pod imenom „digitalizacija“, omogućava da se računarskim putem manipuliše zvučnim zapisima (reprodukovanje, skladištenje), ali isto tako i da se relativno malim memorijskim zauzećem skladište audio zapisi. Kada se kaže „relativno malim memorijskim zauzećem“, misli se na prihvatljiv broj audio zapisa koji „običan“ korisnik može da poseduje na svom memorijskom medijumu.

Kako je problem ovog rada pretraga muzičkih-audio materijala po sadržaju, potreba koja je neminovna je skladištenje ogromnog broja muzičkih numera na memorijskom medijumu, te iz toga proizilazi da je reč „relativno“ u ovom kontekstu, upravo ono što i predstavlja problem. Memorijsko skladištenje nekoliko miliona muzičkih numera, i omogućavanje pretrage po sadržaju, zvuči kao Sizifov[3] posao.

2.1 Svojstva zvuka i fizičke karakteristike

Zvuk čini sastavni deo svakodnevnog života i okruženja u kome se nalazimo. Zvučne oscilacije, odnosno zvuk, nastaje podsticanjem čestica elastične sredine na oscilatorno kretanje i upravo ta sila koja izaziva poremećaj ravnotežnog stanja se naziva izvor zvuka. Oscilujući izvor zvuka koji izaziva promene gustine i pritiska u elastičnoj sredini (vazduh, voda…), generiše zvučne talase u vidu longitudinalnih talasa. Čovečiji „dekriptori“ longitudinalnih talasa su organi sluha i predstavljaju primarni prijemnik zvučnih signala. Ljudski organi sluha su jedni od najosetljivijih čulnih senzora, čiji je primarni zadatak prijem zvučnih podataka i prosleđivanje na obradu ljudskom procesoru (mozak).

Zvuk po svojoj prirodnoj podeli može biti „čist“ i „harmonijski-složen“ zvuk. U prirodi ne postoje „čisti“ zvukovi, čija je karakteristika konstantna frekvencija, već se javljaju kao harmonijsko oscilovanje čestica i takve zvukove nazivamo „harmonijski-složeni“ tonovi, Slika 3. Kao obavezan propratni element harmonijskog zvuka je i „šum“, koji ima sve moguće frekvencijske vrednosti u određenom intervalu. Proučavanje zvuka, odbijanje, apsorpcija, prostiranje, u naučnoj oblasti fizike se naziva „akustika“. Muzička akustika izučava osobine muzičkih tonova i zvukova na osnovu opšte akustike i svojstva organa sluha. Prijem zvučnog signala organima sluha možemo podeliti na *objektivne* i *subjektivne* karakteristike osećaja.



Slika 3. Čisti i harmonijski tonovi

[3] Sizif (grč. Σίσυφος, Sísuphos), najmudriji, najlukaviji i najpokvareniji smrtnik stare Grčke.

2.1.1 Fizičke veličine zvuka (*subjektivne karakteristike*)

Svaki zvuk se sastoji od tonova i prema obliku zvučnog spektra se može podeliti na tri osnovne grupe tonova:

- Prosti tonovi (konstantne frekvencije)
- Složeni-harmonijski tonovi (različite frekvencije)
- Šum (nepravilne frekvencije)

Osnovna karakteristika harmonijskog zvuka, u domenu subjektivnog osećaja, se odlikuje u tome što ih karakteriše veliki broj harmonijskih oscilacija na različitim frekvencijama, koje su sinteza velikog broja prostih zvukova. Jedan složen zvuku može da se sastoji od hiljade prostih zvukova.

Osnovne subjektivne osobine tonova su:

- Visina tona
- Jačina tona
- Boja tona

Visina tona kod složenih-harmonijskih tonova se određuje frekvencijom osnovnog harmonika, a dok kod prostih-čistih tonova je visina tona ujedno i njegova frekvencija. Odavde možemo da zaključimo da je visina tona prvenstveno vezana za frekvenciju. Na visinu tona, takođe utiče i intenzitet zvuka, pa odatle može da se učini da ton malog intenziteta ne daje osećaj visine tona. Ovaj nedefinisan osećaj visine tona se još naziva „*diferencijalni prag čujnosti visine tona*“. Kako važi pravilo da se u svim dijagramima koristi logaritamska skala za frekvenciju, proističe da je visina tona ***proporcionalna logaritmu frekvencije*** [3].

Jačina tona je srazmerna količini energije koju talas prenese kroz jediničnu površinu normalnu na pravac prostiranja talasa u jedinici vremena.

Boja tona je određena relevantnim učešćem viših harmonika u osnovnom tonu [4], odnosno njihovih intenziteta i faznim razlikama osnovnog tona, što predstavlja jedinstvenu osobinu slušnog organa da prepozna dva zvuka iste visine, istog intenziteta (jačine) ali reprodukovana iz dva različita izvora.

2.1.2 Fizičke karakteristike zvuka (*objektivne karakteristike*)

Zvučne talase takođe karakterišu i osnovne fizičke karakteristike, koje još nazivamo „*objektivne karakteristike zvuka*“, i to su: ***osnovna frekvencija, intezitzet zvuka i zvučni spektar***.

Frekvencija zvuka zavisi od talasne dužine zvučnih talasa. Postoji direktna veza između talasne dužine i frekvencije i one su obnuto proporcionalne, a to može da se vidi u formulaciji frekvencije koja kaže, da momenat kada se generiše zvuk i momenta prijema istog, predstavlja talasnu dužinu. Kako je brzina prostiranja zvuka kroz vazduh ≈340m/s, pri temperaturi od 20C°, za jedan frekventni tonski ciklus (1Hz), ton ima talasnu dužinu od 340 metara. Tabela 1 i Tabela 2, prikazuju još neke osobine brzine prostiranja zvuka kroz različite elastične sredine.

Gasovi na 0 C°	**Vazduh**	**Kiseonik**	**Helijum**	**Vodonik**	**CO_2**
Brzina zvuka:	331m/s	316m/s	965m/s	1290m/s	259m/s

Tabela 1. Prostiranje zvuka kroz gaovite elastične sredine

Čvrsta tela:	**Guma**	**Mermer**	**Staklo**	**Čelik**	**Olovo**
Brzina zvuka:	54m/s	3810m/s	5640m/s	5960m/s	1960m/s

Tabela 2. Prostiranje zvuka kroz čvrste elastične sredine

Drugim rečima možemo da kažemo da je frekvencija zvuka, broj ciklusa izraženog u sekundama. Velike talasne dužine karakterišu niske frekvencije, dok su visoke frekvencije opisane malim talasnim dužinama.

Osnovna merna jedinica frekvencije je Herc (eng. *Hertz (Hz)*) [4]. Talasnu dužinu i frekvenciju matematički predstavljamo izrazima:

$$\lambda = \frac{v}{f}$$

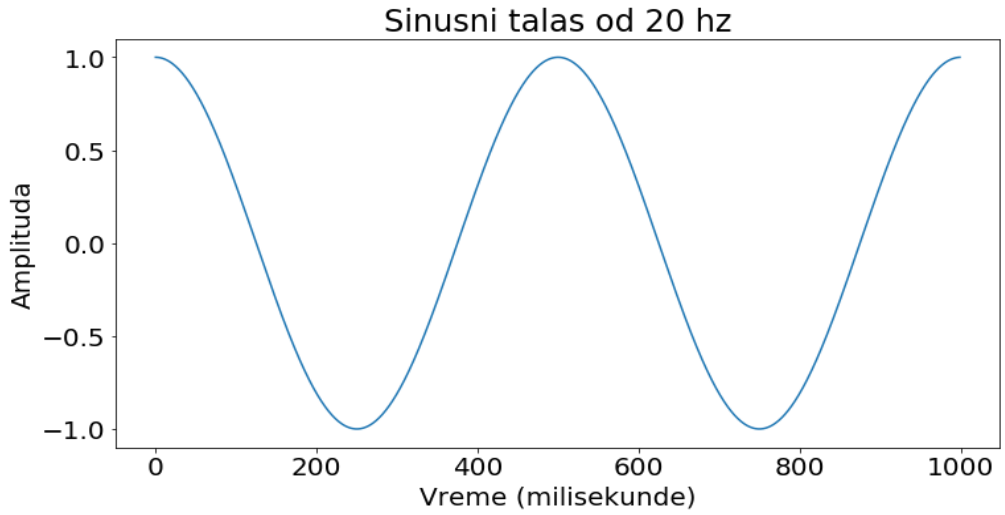
Talasna dužina

$$f = \frac{v}{\lambda}$$

Frekvencija

λ – talasna dužina (m)
v – brzina prostiranja zvučnog talasa (m/s)
f - frekvencija u (Hz)

Intezitet zvuka se još predstavlja i kao njegova amplituda ili veličina frekventnog ciklusa, Slika 4. Za intezitet zvuka je karakteristično da je to mera odstupanja pritiska od srednje vrednosti (stanje tišine).



Slika 4. Karakteristike zvuka – Sinusni talas od 20 hz

[4] Hertz – *Nemački fizičar* ***Heinrich Rudolf Hertz*** (1857–1894)

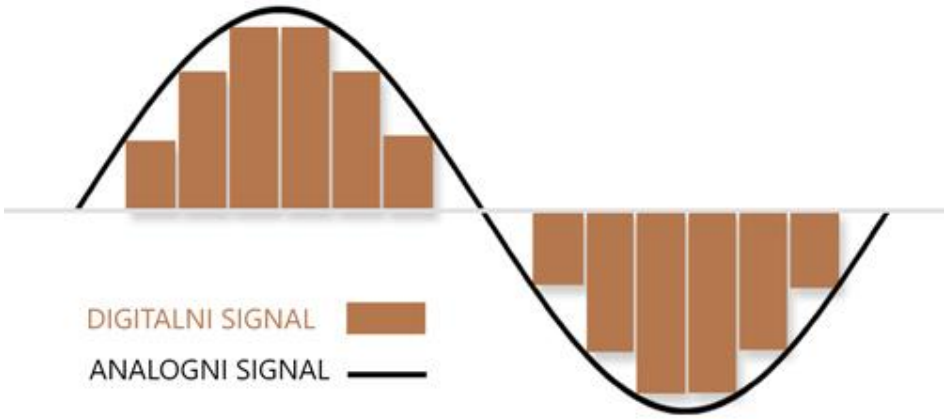
Spektar zvučnih signala, u zavisnosti opsega osnovnih frekvencija, delimo na:

- Infrazvuk: 0-20Hz
- Zvuk koji možemo da čujemo: ≈20Hz – ≈20.000Hz (20kHz)
- Ultrazvuk: 20kHz - 1GHz
- Hiperzvuk: 1GHz – 10THz

Ljudski prijemnici zvučnih signala, organi sluha, mogu da detektuju samo čiste čujne zvukove koji su u frekventnom rasponu od ≈20Hz pa do ≈20kHz. Ovaj frekventni raspon detekcije zvuka se odlikuje svojom degradacijom tokom starenja ljudskog organizma, pa tako starijim osobama respektivno se smanjuje frekventni čujni opseg na ≈15kHz.

2.2 Digitalizacija

Prirodno okruženje ljudske vrste je po samoj svojoj prirodi analogno. Značenje reči analogno u suštinskoj parafrazi predstavlja kontinuum neprekidne vremenske funkcije tj. svi signali koje primamo iz prirode su neprekidni-kontinualni u vremenu i matematički ih možemo predstaviti kao skup neprekidnih vrednosti, odnosno neprekidnom funkcijom. Kako je analogni signal sinonim neprekidnosti u vremenu, za skladištenje analognih signala, u količini koja se generiše danas, bi trebao ogroman vremenski neprekidno rastući medijum za čuvanje na kome bi bile zapisane sve vrednosti signala kako vreme teče. Sigurno da ovaj sistem ne bi funkcionisao jer je nepraktičan i neizvodljiv tehnologijom koju trenutno imamo, jer je dobro poznato da između 0 i 1 postoji beskonačan niz brojnih vrednosti. Iako, ovakva vrsta signala daje perfektan kvalitet reprodukcije, skoro bez gubitka informacija, potreba za skladištenjem i manipulacijom podataka je toliko neophodna, da se morao pronaći kompromis koji balansira između kvaliteta sačuvanih materijala i memorijskog prostora koji je potreban za isti.



Slika 5. Analogni i Digitalni signal

Sa druge strane, pojavljuje se još jedan problem. Pod predpostavkom i da se nađe način kako uskladištiti analogne signale, manipulacija sa takvom vrstom podataka bi bila veoma teška, što dovodi do toga da za izvlačenje informacija iz ovakvih sirovih-analognih podataka, bi bio potreban i kopatibilan računar-mašina, koja je u osnovi analogna, drugim rečima, to bi bio analogni računar sa beskonačnom memorijom.

Na „sreću“, nesavršenstvo ljudskog organa sluha daje mogućnost da analogni signal digitalizujemo, Slika 5, a kao posledica toga je gubitak informacija signala. Ovaj gubitak informacija usled nesavršenstva sluha se ni ne primećuje.

Predstavljanje analognog signala nizom brojeva u diskretnim vremenskim intervalima, se naziva ***digitalizacija***. Uređaj koji pretvara analogni signal u digitalni se zove A/D konverter (eng. *ADC–analog-to-digital converter*). On meri jačinu analognog signala u diskretnim vremenskim intervalima i pretvara izmerene vrednosti u digitalne (binarne) brojeve, tj. dodeljuje svakoj tački neku vrednost iz skupa diskretnih vrednosti (nivoa kvantizacije), koja je najbliža stvarnoj vrednosti signala.
Rezultat beleženja vrednosti analognog signala u određenim vremenskim intervalima, tj. u tačkama vremena, nazivamo ***diskretizacija signala***, pa kako se računarska obrada signala svodi na rad sa brojevima kao predstave tog signala, one moraju da uzimaju vrednosti iz diskretnog skupa. Takav signal se još naziva ***digitalni signal***. Postupak pri konverziji analognog signala u digitalni se može predstaviti u tri koraka:

1. Odmeravanje (eng. *Sampling*)
2. Kvantizacija (eng. *Quantizing*)
3. Kodovanje (eng. *Encoding*)

2.2.1 Odmeravanje – diskretizacija po vremenu

Podaci koji se prenose analognim signalima su vremensko-amplitudno kontinualni i sadrže informacije koje su, ukoliko se zanemari šum, neoštećene, tj. maksimalno je iskorišćen informacioni kapacitet kanala prenosa. Da bi se takav signal diskretizovao, a to je neophodno za manipulaciju na računarima, potrebno je izvršiti odmeravanje signala. Negativan nusprodukt odmeravanja je gubitak informacija u signalu, i ukoliko je potreba za potpunom rekonstrukcijom digitalnog signal u analogni, mora da se pazi na brzinu odmeravanja.

Potpuna upotrebna snaga digitalizacije zvuka je upravo u što vernijoj, originalnijoj rekonstrukciji diskretnog signala u analogni signal. Kako bi to uspešno uradili, potrebno je matematičko izvođenje zasnovano na osobinama Furijeove transformacije i uslov koji mora biti ispunjen poznat je pod *teoremom odmeravanja* [3].

$$\boldsymbol{T_0 = \frac{1}{f_0} \leq \frac{1}{2f_g}} \qquad (1)$$

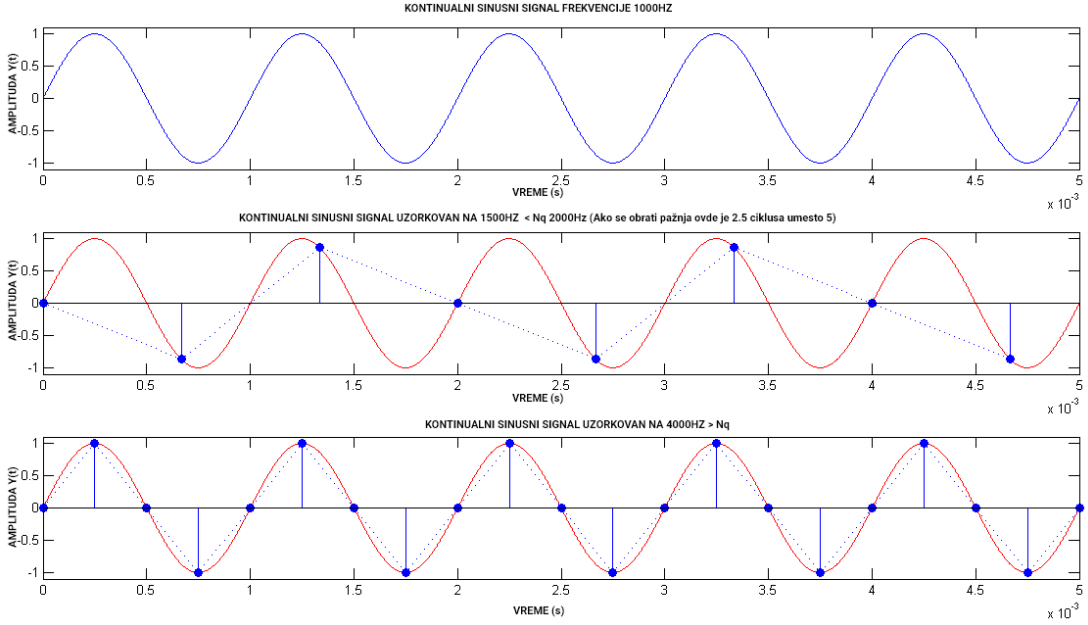
Gde je u izrazu (1), T_0 period odmeravanja, f_0 frekvencija odmeravanja, a f_g granična frekvencija u spektru audio signala. Kako ljudski organi sluha registruju samo frekventni raspon od ≈20Hz pa do ≈20kHz , a prema ***Nyquist–Shannon*** [5] teoremi minimalne frekvencije odmeravanja (1), ako želimo da uzorkujemo signal od 20kHz potrebno je minimalno 40,000 uzoraka po sekundi. Vremenski standard odmeravanja koji se danas koristi na Audio CD-u, MP3, je 44,1 kHz ,odnosno 44,100 samplova u sekundi.

Standardi koji se najčešće koriste za uzorkovanje na određenim uređajima su prikazani u Tabela 3.

PLATFORMA	UZORKOVANJE	KVANTIZACIJA	PROTOK (Kb/sec)	FREKVENCIJA
Telefon	8 kHz	8 bit	8 Kb/sec	0,200-3,4 kHz
AM/radio	11,025 kHz	8 bit	11,0 Kb/sec	0,1-5,5 kHz
FM/radio	22,05 kHz	16 bit	88,2 Kb/sec	0,02-11 kHz
CD	44,1 kHz	16 bit	176,4 Kb/sec	0,005-20 kHz
DAT	48 kHz	16 bit	192,0 Kb/sec	0,005-20 kHz
DVD Audio	192 kHz (max)	24 bit (max)	1200,0 Kb/sec	0-96 kHz (max)

Tabela 3. Najčešći standardi uzorkovanja analognih signala

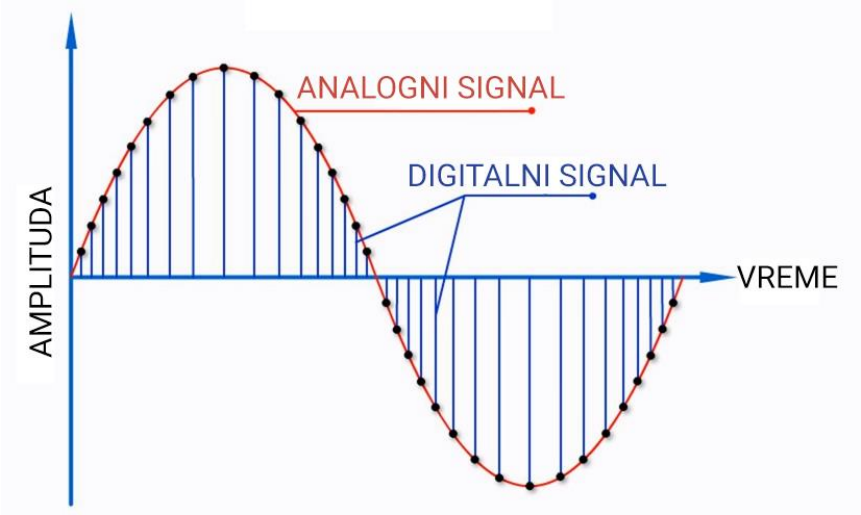
Pošto većina analognih signala koji dolaze iz stvarnog sveta nemaju ograničen frekvencijski spektar, postoji mogućnost da se pojavi preklapanje spektra (eng. *aliasing*), tako da prilikom uzimanja uzoraka mora da se poštuje pomenuta *Nyquist–Shannon* teorema, koja dokazuje da je za idealnu rekonstrukciju originalnog signala potrebno da frekvencija odmeravanja bude bar dva puta veća od najviše frekvencije, tj. da imamo barem dve tačke po frekvencijskom ciklusu, Slika 6.



Slika 6. Primer dobrog i lošeg odmeravanja prema Nyquist–Shannon teoremi

Takođe, da bi se izvršila minimalizacija pojave preklapanje spektra, pre svakog A/D konvertera se postavlja niskopropusni filter (eng. *Antialiasing filtar*).

Postavljanje ovog filtera osigurava slabljenje osnovnih harmonika na frekvencijama višim od polovine frekvencija odmeravanja. Pošto ljudski sluh slabo percipira visokofrekventne signale približavanjem granici čujnosti od ≈20kHz, gubitak informacija, koji se implementacijom ovog filtera dobija, je potpuno prihvatljiv jer je poznato da energija realnog signala opada sa porastom frekvencije.

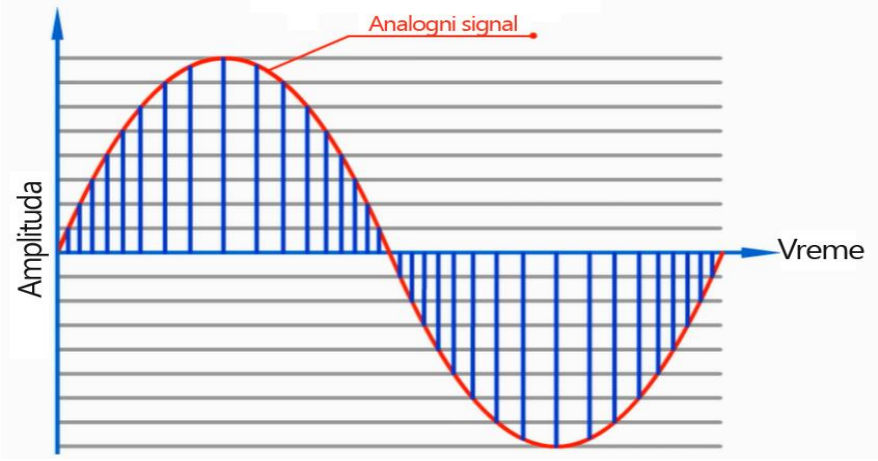


Slika 7. Diskretizacija signala po vremenskoj (x) osi (vertikalno odmeravanje)

Obzirom da se iz matematičkog prikaza teoreme odmeravanja vidi diskretizacija po vremenskoj (horizontalnoj) osi, ovakav vid odmeravanja se još naziva „*vertikalno odmeravanje*“, kao što je prikazano na Slika 7.

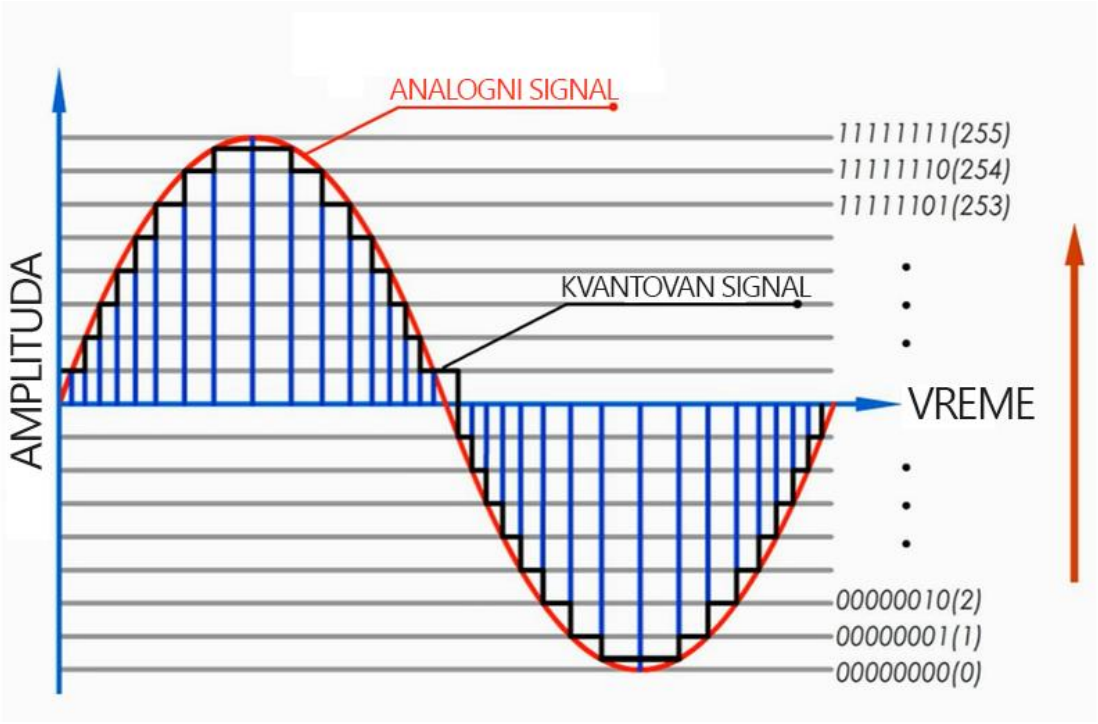
2.2.2 Kvantizacija – diskretizacija po amplitudi

Prilikom procesa diskretizacije audio signala, da bi se došlo do pravilnog odmeravanja signala, fokus je bila vremenska osa (x-osa), dok kod procesa kvantizacije fokus uzima vertikalna osa, tzv. amplitudna osa (y-osa). Vizuelno se to vidi na Slika 8.



Slika 8. Diskretizacija po amplitudnoj (y) osi (horizontalno odmeravanje)

Pretvaranje kontinualnih vrednosti u diskretne, na takav način da se vrednosti mernih signala zaokružuju na najbliži celobrojni iznos, se naziva proces kvantizacije. Svaki kontinualan signal nakon diskretizacije po vremenu (odmeravanje), je sačinjen od N delova, gde svaki deo ima vrednost kontinualnog intervala (A_{min}, A_{max}), gde A predstavlja amplitudalni signal u digitalnom obliku. Međutim, ukoliko bi želeli takav signal da predstavimo u digitalnom obliku (bez kvantizacije), došli bi u situaciju da bi svaki segment bio predstavljen sa beskonačno mnogo tačaka, kodovan beskonačno dugom kodnom rečju iz opsega vrednosti (A_{min}, A_{max}). Upravo zbog te osobine, pre kvantizacije signala po amplitudnoj osi (y-osa), on je još uvek kontinualan, pa u procesu kvantizacije signala neminovno dolazi do gubljenja informacija. tj. gube se međuvrednosti koje su bile prisutne pre kvantizacije. Kako je naglašeno, kvantizacija je ireverzibilan proces, tj. nemoguće je egzaktno rekonstruisati analogni signal i povratiti sve informacije nakon kvantizacije. Slika 9, prikazuje primer kvantovanog analognog signala u 8-bitnoj preciznosti sa 256 nivoa.



Slika 9. Kvantovanje analognog signala u 8-bitnoj rezoluciji sa 256 nivoa

Kako je naglašeno da amplitudalni signal može da poprimi vrednosti iz opsega (A_{min}, A_{max}), dinamički opseg signala bi se predstavio kao $A_{max} - A_{min}$. Ako podelimo ovaj interval na N kvantizacionih nivoa $A_0 = A_{min} < A_1 < A_2 < \ldots < A_{N-1} = A_{max}$. Izraz:

$$\Delta = \frac{A_{max} - A_{min}}{N - 1}$$

predstavlja rezoluciju ili korak kvantizacije. Broj kvantizacionih nivoa je povezan sa brojem bita B, potrebnih za kodovanje i vredi da je $N = 2^B$, Slika 9. Respektivno, zaključuje se da što je korak kvantizacije veći (grublja kvantizacija), to je potrebno manje bitova za kodovanje odmeraka. Analogno, što je korak kvantizacije manji (finija kvantizacija), to je potrebniji veći broj bitova za kodovanje odmeraka.

Standardna preciznost odmeravanja je 2 bajta, tj. 16 -bitna, kako bi bio rešen problem dinamike zvuka, a to je odnos između najtišeg i najglasnijeg zvuka. Naime, raspon jačine zvukova u prirodi koje ljudski organi sluha mogu da registruju je izuzetno veliki[5].

Priroda je rešila ovaj problem tako što ljudske uši čuju logaritamski:

- zvuk koji se čuje, a dva puta je jači od nekog drugog zvuka, zapravo nije duplo jači nego $10^2 = 100$ puta jači.
- Zvuk koji se čuje tri puta jače, je $10^3 = 1000$ puta jači.
- Uopšte za ljudski organ sluha, $\boldsymbol{N}$ puta jači zvuk je u stvari je 10^n puta jači.

Dakle, ljudski organi sluha su vrlo osetljivi i pokrivaju ogroman raspon jačina zvuka, te iz tog razloga je potrebna najmanje 2 -bajtna:16 -bitna preciznost koja omogućava 256 = 65.536 kvantizacionih nivoa vertikalne diskretizacije, kako bi i digitalno snimljen zvuk takođe pokrivao ovaj veliki raspon, ne bi li kvalitet reprodukcije bio na zadovoljavajućem nivou.

Proces kvantizacije je ireverzibilan proces i nakon izvršene kvantizacije više nije moguće rekonstruisati originalan signal. Ovo je upravo i pokazatelj da se u procesu kvantizacije unosi i greška koja se najčešće naziva *greška kvantizacije* ili *šum kvantizacije*[6]. Šum kvantizacije je razlika između kvantovane i originalne vrednosti [6]:

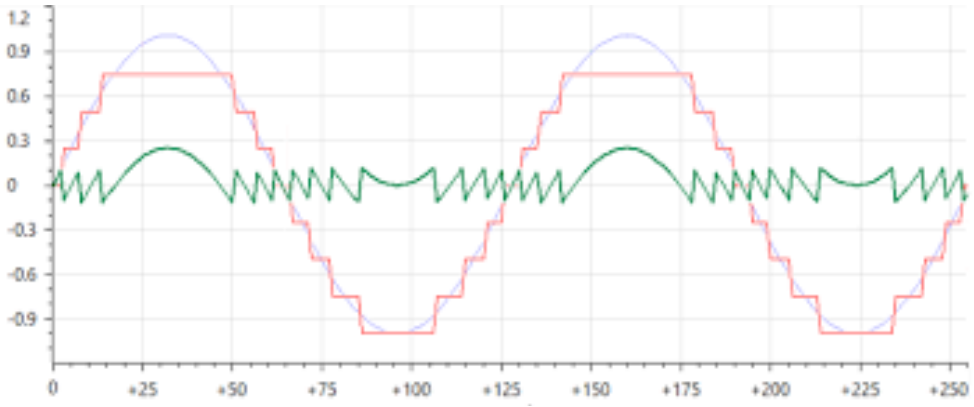
$$\mathbf{e_q(n) = x_q(n) - x(n)}.$$

Uticaj šuma na kvalitet digitalizovanog audio signala se može matematički prikazati objektivnošću odnosa signal-šum i to matematičkim izrazom:

$$\boldsymbol{10log\frac{Px}{Pn}\ [dB]},$$

dok se sam kvalitet kvantovanog audio signala može izraziti pomoću odnosa signal-šum kvantizacije (eng. *Signal-to-Quantization-Noise Ratio, SQNR*), Slika 10:

$$\boldsymbol{SQNR = 10\log\frac{P_x}{P_q}\ [dB]}$$



Slika 10. Šum kvantizacije

[5] Odnos jačine najtišeg i najglasnijeg zvuka je oko $1 : 10^9$.

[6] Šum je po prirodi slučajna vrednost, a kako je greška kvantizacije slučajna, otud naziv *šum kvantizacije.*

2.2.3 Kodovanje – Linearna impulsna kodna modulacija

Impulsna kodna modulacija je jedan od standarda za predstavljanje digitalnih signala. Patentirao ga je *A. Reeves*[7] 1930-te godine. Upotreba ove modulacije je našla svoju primenu u većini elektronskih uređaja, poput kompakt diskova, transformacije MP3 signala u PCM signal, koji se zatim šalje u slušalice ili zvučnik.

Postoji više formata PCM signala, ali najčešće korišćen je onaj u audio signalima i to na 44.1 kHz, 16 -bitnom dubinom, stereo formata. Ovakav format se sastoji od 44.100 uzoraka svake sekunde muzike, dok svaki uzorak ima 4 bajta (32 bita), Slika 11:

- 2 bajta (16 bita) za intenzitet od -32.768 do 32.767 levog zvučnika
- 2 bajta (16 bita) za intenzitet od -32.768 do 32.767 desnog zvučnika

Slika 11. PCM – 16-bitno stereo kodovanje

[7] ***Alec Harley Reeves*** *(10. mart 1902. – 13. Oktobar 1971.)* Britanski naučnik i pronalazač.

3. Priprema digitalnog audio signala za pretraživanje

Osnovna polazna tačka svakog sistema za pretragu je spoznaja jedinstvenosti u predmetu pretrage. Ukoliko želimo nešto da pronađemo, prvo i pre svega moramo da znamo kako „*to*“ izgleda, da znamo jedinstvene osobine koje predmet pretrage ima. Ukoliko je pretraga po malom uzorku, te jedinstvenosti mogu da budu manje, dok ukoliko pretražujemo u uzorku koji poseduje nekoliko miliona ili milijardi jedinki, jedinstvenosti moraju da budu velike i unikatne.

Drugim rečima, jedinstvenosti predmeta pretrage po velikom uzorku je obrnuto proporcionalna jedinstvenostima pretrage po malim uzorcima. Kako je ljudski rod premašio brojku od $7 * 10^9$, potreba za brzom identifikacijom jedinki ljudskog roda je bila neminovna.

Da bi se to postiglo, moralo se doći do jedinstvenih osobina svake jedinke ljudskog roda, a to su biometrijske osobine čoveka. Otisak prsta je jedinstven za svaku osobu ljudskog roda, retina u čovečijem oku, itd. Kada imamo ovako jedinstven podatak za svaku jedinku, informacija za pretragu i identifikaciju, uz malu pomoć matematičkih funkcija, ne predstavlja problem.

Vođeni postupkom prepoznavanja jedinke ljudskog roda putem biometrijskih podataka, možemo da „imitiramo“ i sa audio zapisima. Pronalaženje jedinstvenog „otiska prsta“ muzičkog zapisa, daje mogućnost pretrage muzičkih materijala po sadržaju. U narednih nekoliko poglavlja ćemo se usredsrediti na postupak izvlačenja ovih informacija iz audio sadržaja i njihovo skladištenje za kasniji postupak upoređivanja.

3.1 Spektrogramski otisak zvuka

Svetlosni snop je deo elektromagnetnog zračenja iz opsega talasnih dužina vidljivih golim okom i kao takav sastavljen je od talasa u rasponu od 380nm (ljubičasta boja) pa sve do 750nm (crvena boja). Ljudski očni organi vide svetlost kao belu boju, međutim kada se izvrši disperzija zraka bele svetlosti kroz staklenu prizmu, dolazi do razdvajanja dužnih talasa koji se vide u prirodi kao dugine boje [7].

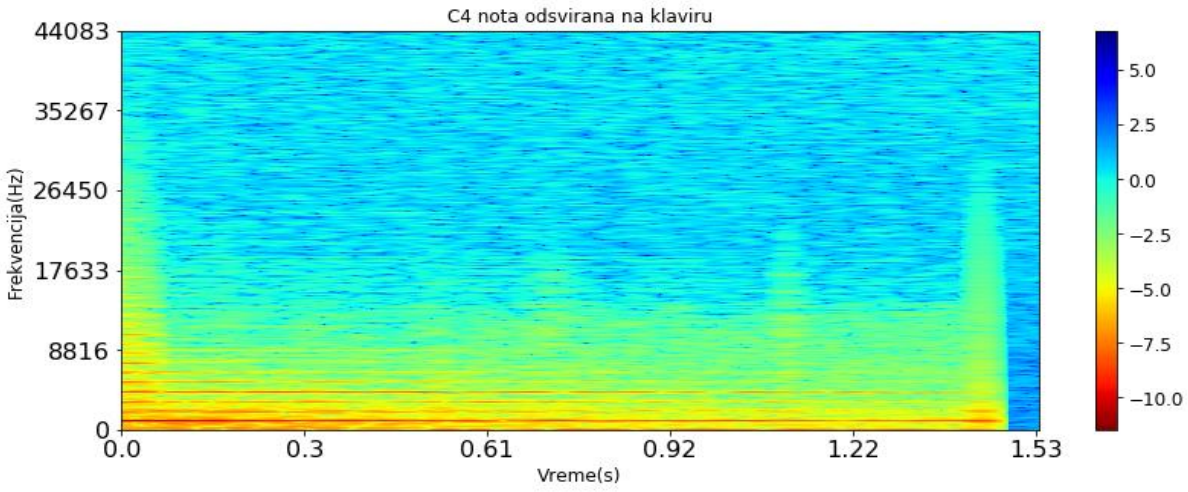
Kada se ovo svojstvo svetlosti uporedi sa zvučnim talasima, nesumnjivo se vidi da postoji izvesna sličnost u „materijalnoj kompoziciji“ i da ono što vidimo i čujemo je ustvari skup kompleksnijih talasa. Zvukovi koji dopiru do ljudski organi sluha su složeni zvučni signali, a oni su sastavljeni od jednofrekventnih „čistih“ zvukova.

Ljudski dekriptori zvučnih signala se ponašaju slično kao „Furijerov[8] analizator“ i razdvajaju kompleksne zvučne talase na spektar jednostavnih talasa. Ova osobina daje jedinstvenu mogućnost da npr. lako prepoznamo nečiji glas, jer svaka ličnost ima posebnu boju, tonalitet glasa, koja ostavlja jedinstveni spektralni otisak. Ovaj, na izgled jednostavan zadatak prepoznavanje glasa i povezivanje sa određenom ličnošću, nama ne predstavlja preveliki problem i nekako to ‘intuitivno radimo[9], bez prevelikog napora, dok implementacija ovog istog zadataka i prebacivanje u digitalan svet, zahteva preplitanje više informatičko-matematičkih oblasti koje nesumljivo vode do velike kompleksnosti.

[8] ***Jean-Baptiste Joseph Fourier*** – Francuski matematičar i fizičar.

[9] Kada se čuje poznati zvuk, do prepoznavanja dolazi kada se u ljudskom mozgu aktiviraju određeni neuroni, koji opet otključavaju deo mozga gde je informacija uskladištena.

Spektogramski otisak zvučnog signala predstavlja trodimenzionalni graf, koji u svojoj osnovi oslikava kombinaciju osnovnih komponenti frekvencija, iz kojih se može izvesti originalan signal i koliko su pojedine frekvencijske komponente imale udeo u originalnom signalu, Slika 12.



Slika 12. Spektralni otisak tona C, odsviran na klaviru

Sama mogućnost prikaza spektra i njegova analiza je od velikog značaja, jer daje mogućnost selektivnog filtriranja frekvencija sa kojima se kasnije može manipulisati.

Selektivna manipulacija frekvencijama se odnosi na diskretne signale koji se nalaze samo u digitalnom obliku, dok kod kontinuiranih-analognih signala ova mogućnost je veoma teška pa skoro i nemoguća, što ukazuje da pre nego što se izvrši bilo kakva manipulacija nad audio signalima, potrebno ga je digitalizovati, postupak koji je detaljno opisan u poglavlju 2. „*Teorijske osnove digitalizacije zvuka*“. Dobijanje sprektralnog otiska se može uraditi analitički, primenom Furijerove transformacije ili eksperimentalno, korišćenjem analizatora spektra.

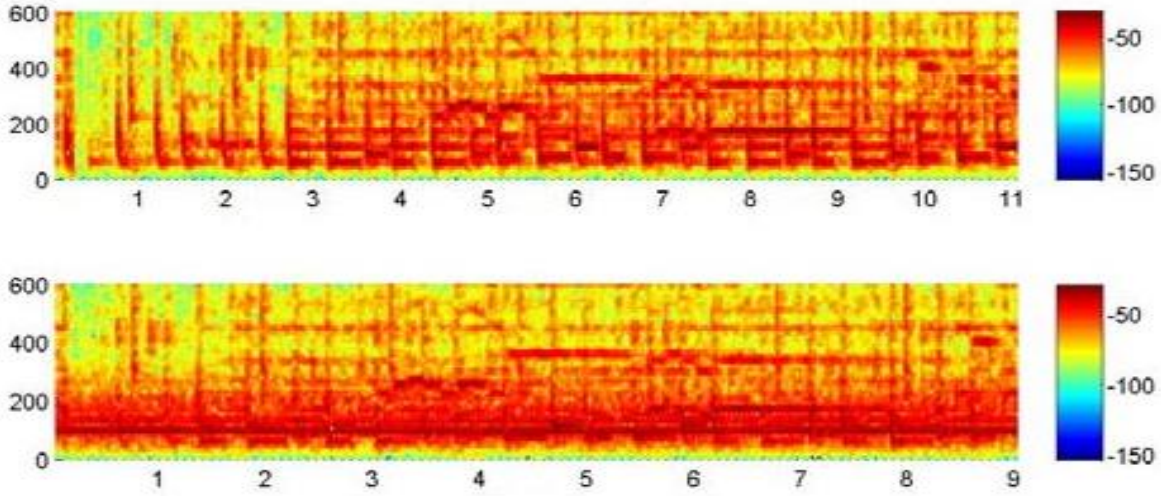
3.2 „Otisak prsta“ audio signala

Kada se kaže „audio otisak prsta (eng. *Audio fingerprint*)[10]“, misli se na kompaktnu reprezentaciju nekog audio signala koji može da bude muzika, zvuk iz okruženja itd. a enkapsulira informaciju koja je specifična samo za taj zvuk koji ga predstavlja. Svrha uzimanja audio otiska je da se određeni podatak zvuka distancira od ostalih zvukova. Važno je napomenuti, da je jedinstveni otisak audio signala usko povezan za instancu koja se upoređuje, a ne za klasu zvukova kao što su npr. zvuk kiše, zvuk automobila, zvuk vetra.

Danas postoje razni servisi koji koriste tehnologiju uzimanja audio otiska, a neki od njih su: *Shazam*, *AcoustID*, itd. i svima je zajedničko da su bazirani na ekstrakciji otiska iz vremensko-frekventnog domena audio signala, koji se još naziva i spektrogram.

[10] Ponekad se kaže i akustičan otisak prsta (eng. *Acoustic fingerprint*).

Audio sprektrogrami su odlična polazna tačka, jer dozvoljavaju identifikaciju vremensko frekventnih sadržaja i intenzitet svake frekvencije u signalu. Spektrogram sam po sebi ne znači mnogo, jer za ekstrakciju jedinstvenosti jednog audio signala potrebno je smanjiti obim podataka na minimum. Sirov spektrogram poseduje previše redundantnih podataka („*šum*“ okoline) koji degradiraju kvalitet audio signala, što se vidi na Slika 13.



Slika 13. Audi signal bez šuma(gornja slika) i sa šumom(donja slika)

Slika 13. prikazuje sirovi (eng. *raw*) spektrogram dva identična zvuka ali snimljena u dva različita okruženja. Gornja slika signala je bez pozadinskog šuma, dok je donji signal snimljen u okruženju koji ima pozadinski šum. Iako se vidi da su to skoro dva različita spektrograma, jedno im je ostalo skoro netaknuto, a to su vrhovi (eng. *peak*). Upravo ti vrhovi su odlična polazna tačka za kreiranje jedinstvenog audio otiska prsta.

Jedinstveni otisak koji daje audio signal spektralnom analizom je upravo ono što je potrebno kao „*biometrijsko*“ svojstvo audio signala, a za potrebu pretrage i identifikacije predmeta pretrage. Servisi [8], [9], [10], koji su ranije spomenuti, koriste ovaj pristup za identifikaciju audio materijala. Iako ovi servisi imaju patentirane algoritme za izvlačenje audio otiska, objavljeni su naučni radovi na temu [1] [11], iz kojih se vidi da su u globalu koristili metode uzimanja audio spektralnog otiska, kao osnovu implementacije svojih sistema .

Kako je ovo najpopularniji i „najočigledniji“ način za proces identifikacije audio materijala, baziran na generisanju jedinstvenih tačaka u audio signalu, očigledna je primena procesa transformisanja važnih tačaka frekventnih informacije u signalu. Dobijanje frekventnih informacija iz analognih audio signala je matematička funkcija francuskog matematičara „*Jean-Baptiste Joseph Fourier*“ i poznata je pod imenom „Furijerova transformacija“, koja vremensko frekventne signale razlaže na frekvencije koje ga čine. Signali sa kojima se radi u multimedijalnim sistemima su digitalni, pa je Furijerovu transformaciju potrebno prilagoditi diskretnom domenu.

3.2.1 Frekvencijska predstava digitalnog signala

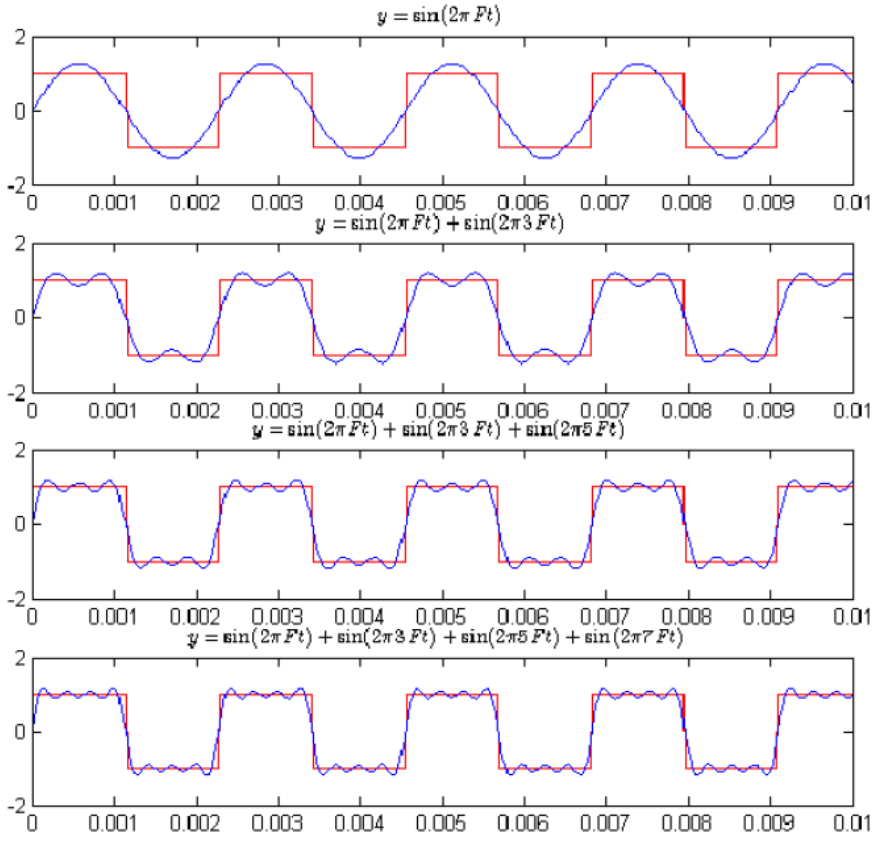
Za određivanje frekvencija u kontinualnom analognom signalu se koristi matematička funkcija pod nazivom „*Furijerova transformacija*“. Ideja francuskog matematičara *Žana Batista Furijea* se sastoji u tome da je, periodične funkcije moguće razložiti na linearnu kombinaciju prostoperiodičnih (sinusnih) funkcija [12]. Ova reprezentacija periodičnih funkcija se naziva Furijerov red, Slika 14, ima oblik:

$$x(t) = \frac{a_0}{2} \sum_{k=1}^{\infty} (a_k \cos(k\Omega_0 t) + b_k \sin(k\Omega_0 t)) \qquad (2)$$

gde su a_k i b_k koeficijenti Furijeovog reda, a dobijaju se kao:

$$a_k = \frac{2}{T} \int_0^T x(t) \cos(k\Omega_0 t) dt \qquad (3)$$

$$b_k = \frac{2}{T} \int_0^T x(t) \sin(k\Omega_0 t) dt \qquad (4)$$



Slika 14. Aproksimacije pravougaonog signala Furijeovim redom

Obzirom da se gravitira u domenu diskretnih signala, potrebno je iskoristiti alate pod Furijerovim transformacijama. Drugim rečima, ako se primeni Furijerova transformacija nad analognom audio signalu, kao rezultat se dobijaju frekvencije i njihov intezitet unutar tog audio signala, dok upotrebom matematičke funkcije pod nazivom „*Diskretna Furijerova Transformacija*“ (*eng. Discrete Fourier Transform,* DFT), se dobija mogućnost određivanja frekvencija i njihovog inteziteta nad digitalnim audio zapisima. DFT ima mala ograničenja na koja se mora posebno obratiti pažnja, jer transformacija DFT funkcije se izvršava samo na jednokanalnom uzorku, tj. ako je stereo (dvokanalni) digitalni audio zapis, pre implementacije DFT transformacije, potrebno je izvršiti transformaciju u jednokanalni (mono) zapis. Upotreba DFT transformacije daje diskretan spektrum frekvencija i njihov intezitet koji je opisan opštim matematičkim izrazom (5):

$$X(k) = \sum_{n=0}^{N-1} x_n e^{\frac{-j2\pi kn}{N}} \tag{5}$$

Gde je:

- **N** veličina „prozora“: broj uzoraka kojim je konstruisan diskretni signal
- **X(k)** predstavlja **k**-ti frekvencijiski impuls(bin)[11]
- **X(n)** predstavlja **n**-ti uzorak audio signala

Usled razvijanja kompleksong eksponencijala posle $-\boldsymbol{j}$ u (5) se zamenjuje ostatak eksponencijala sa $\boldsymbol{b_n}$ pa je $\frac{\boldsymbol{2\pi kn}}{\mathbf{N}} \leftrightarrow \boldsymbol{b_n}$. Nakon razvijanja sume u (5), se dobija:

$$X(k) = x_0 e^{-jb_0} + x_1 e^{-jb_1} + \dots\ x_n e^{-jb_{N-1}} \tag{6}$$

Ukoliko se zna da Ojlerovom formulom kompleksnih brojeva može da se predstavi kompleksni eksponent sa: $\boldsymbol{e^{jx} = \cos x + j \sin x}$ dobija se niz sinusnih i kosinusnih vrednosti:

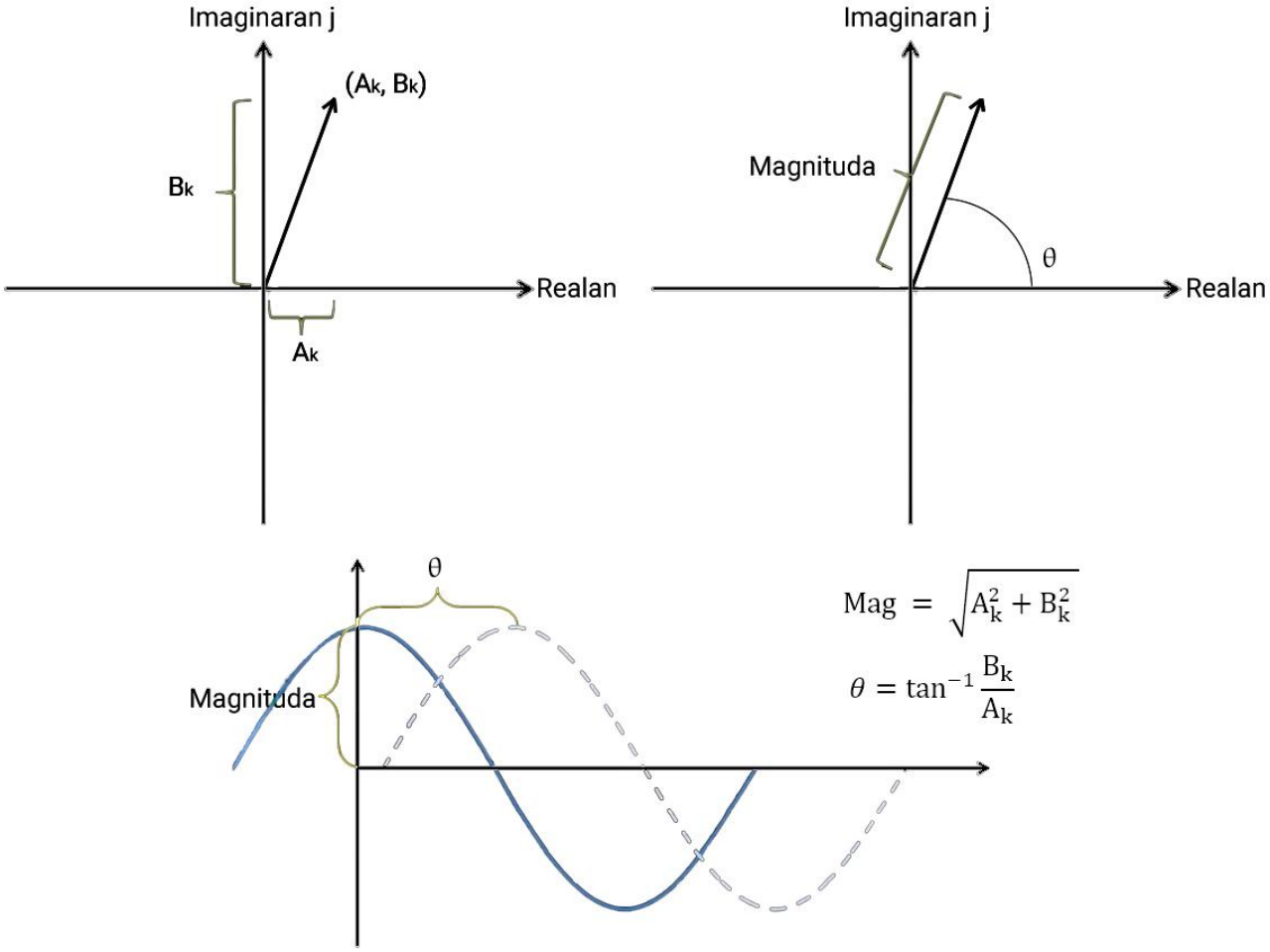
$$X(k) = x_0[cos(-b_0) + j\sin(-b_0)] + \cdots \tag{7}$$

Nakon sumiranja u (7) se dobija konstantan kompleksni broj oblika:

$$X(k) = A_k + B_k j \tag{8}$$

[11] ***Frekvencijiski binovi*** su intervali između uzoraka u frekventnom domenu.

Dobijeni kompleksan broj u (8), može da se predstavi u kompleksnom koordinatnom sistemu kao vektor sa jednom imaginarnom koordinatom i jednom realnom koordinatom kao što je prikazano na Slika 15.



Slika 15. Grafički prikaz dobijenog kompleksnog broja iz DFT-a

Izračunavanja amplitude audio signala se radi Pitagorinom teoremom kao u (9), dok se pomeraj (*eng. shifting*) radi preko tangensa kao u (10)

$$Mag = \sqrt{A_k^2 + B_k^2} \tag{9}$$

$$\theta = \tan^{-1}\frac{B_k}{A_k} \tag{10}$$

Za efkasno izračunavanje DFT se koristi algoritam pod nazivom „*Brza Furijeova transformacija"*(*eng. Fast Fourier Transform,* FFT*).* Danas postoje efikasne implementacije FFT algoritma u obliku programskih biblioteka, koje se mogu koristiti u različitim programskim jezicima.

Brza Furijeova transformacija je algoritam za "brzo" izračunavanje vrednosti diskretne Furijeove transformacije. Ubrzanje u odnosu na uobičajan postupak izračunavanja diskretne Furijeove transformacije se postiže izbegavanjem ponovnog izračunavanja izraza koji se međusobno negiraju. Algoritam se pripisuje *Džejmsu V. Kuliju* (*James W. Cooley*) i *Džonu V. Tukiju* (*John W. Tukey*) kojeg su ga objavili 1965. Godine [13].

Međutim, *Karl Fridrih Gaus* ga je razvio još 1805. godine, da bi izračunao putanju asteroida Palas i Juno. Pri tom su mnoge verzije razvijene i pre Kulijeve i Tukijeve varijante.

3.2.2 Prozorske funkcije

Najčešći problemi prilikom implementacije DFT koristeći FFT algoritam je, efekat curenja spektra (eng. *Spectral leakage*). Efekat curenja spektra se neminovno mora desiti, jer se energija signala „razliva" u širokom opsegu frekvencija oko glavnih harmonika, umesto da se nalazi u ograničenom frekventnom opsegu neperiodičnog diskretizovanog signala. Curenje spektra se ne može izbeći, ali odabirom odgovarajuće prozorske funkcije (eng. *window function*), može da se utiče na ponašanje curenja spektra.

Prozorska funkcija aproksimira originalni audio signal i ispravlja efekte curenja spektra i tokom vremena su predloženi razni „prozori", noseći sa sobom svoje prednosti i nedostatke. Neke od prozorskih funkcija su:

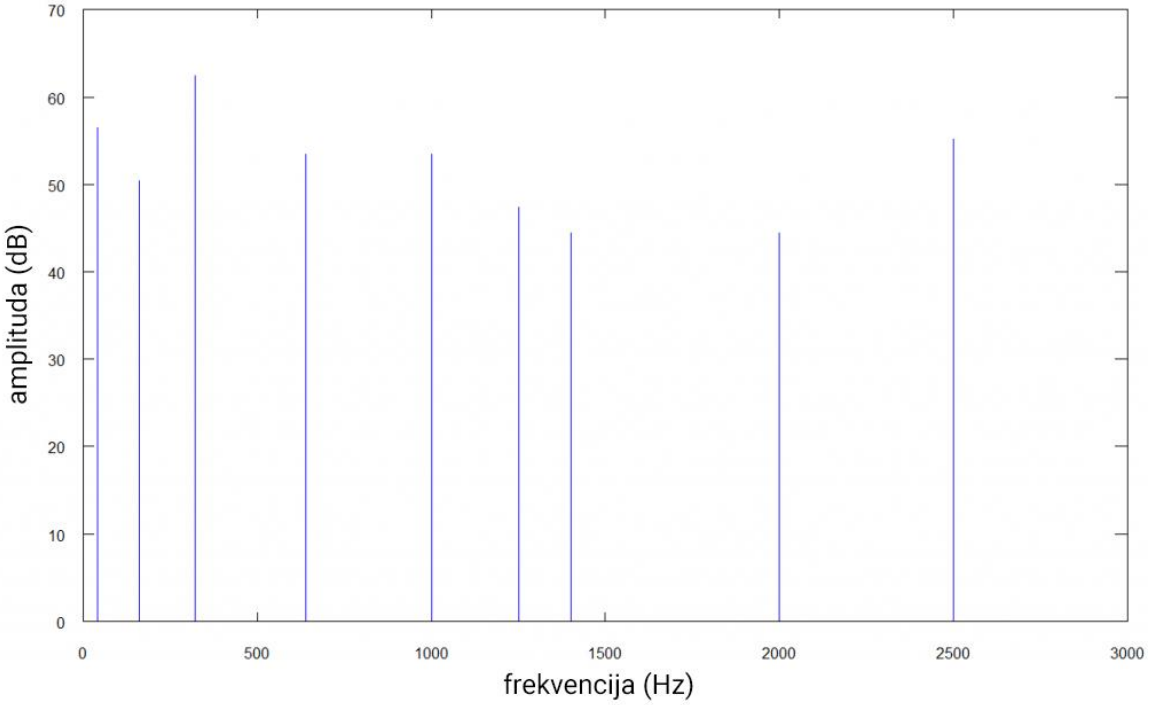
- „*Pravougaoni*" prozor
- „*Hammingov*" prozor
- „*Hanningov*" prozor
- „*Bartlettov*" prozor
- „*Gaussov*" prozor
- „*Blackmanov*" prozor

Određene prozorske funkcije su bolje u poboljšavanju frekventnih rezolucija, tj. olakšavaju tačnije određivanje centralne frekvencije, dok su se druge pokazale bolje u određivanju merenja amplitude na centralnoj frekvenciji.

Predstavićemo primere iz domena prozorskih funkcija i to za: „*Pravougaonu*", „*Blackmanovu*" i „*Hammingovu*" funkciju. Predpostavimo da imamo audio signal sačinjen od:

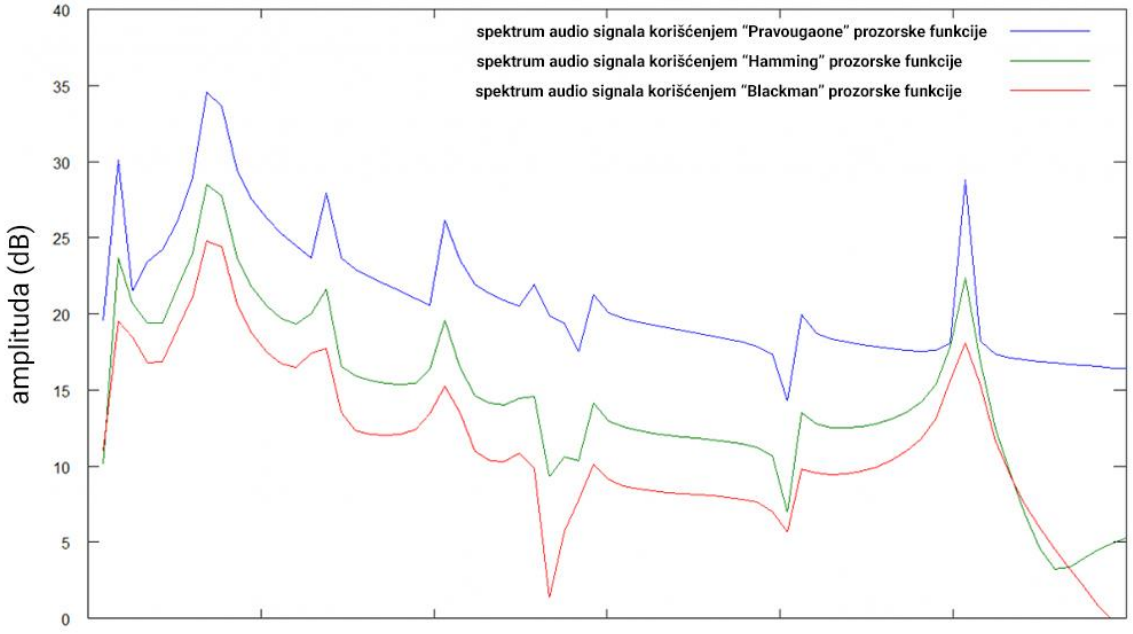
- Frekvencija **40Hz** sa amplitudom **2**
- Frekvencija **160Hz** sa amplitudom **0.5**
- Frekvencija **320Hz** sa amplitudom **8**
- Frekvencija **640Hz** sa amplitudom **1**
- Frekvencija **1000Hz** sa amplitudom **1**
- Frekvencija **1225Hz** sa amplitudom **0.25**
- Frekvencija **1400Hz** sa amplitudom **0.125**
- Frekvencija **2000Hz** sa amplitudom **0.125**
- Frekvencija **2500Hz** sa amplitudom **1.5**

U savršenom prikazu Furijerove transformacije bi se dobio spektrum kao na Slika 16.



Slika 16. Prefektan spektrum audio signala

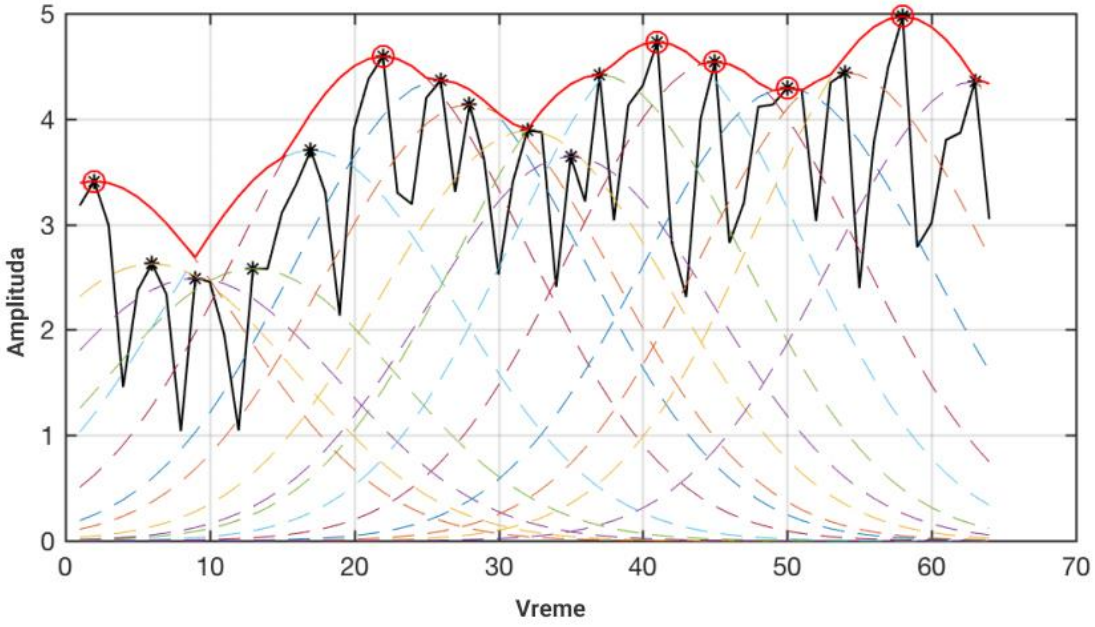
Za dobijanje ovako savršenog spektrograma potrebno je primenuti Furijerovu transformaciju na prozorskoj funkciji koja je veoma dugačka (10 sekundi). Ukoliko se primeni ovako dugačka prozorska funkcija, curenje spektra je veoma mala, ali pošto su promene u audio signalima veoma brze, ovakav pristup je neprihvatljiv. Umesto primenjivanja dugačke prozorske funkcije, potrebno je podeliti signal na veoma male delove, reda veličina 0.1 sekunde. Spektrum nakon 4096 prozorskih uzorkovanja izgleda kao na Slika 17.



Slika 17. Furierova transformacija signala 44.1kHz sa 1024 prozorom

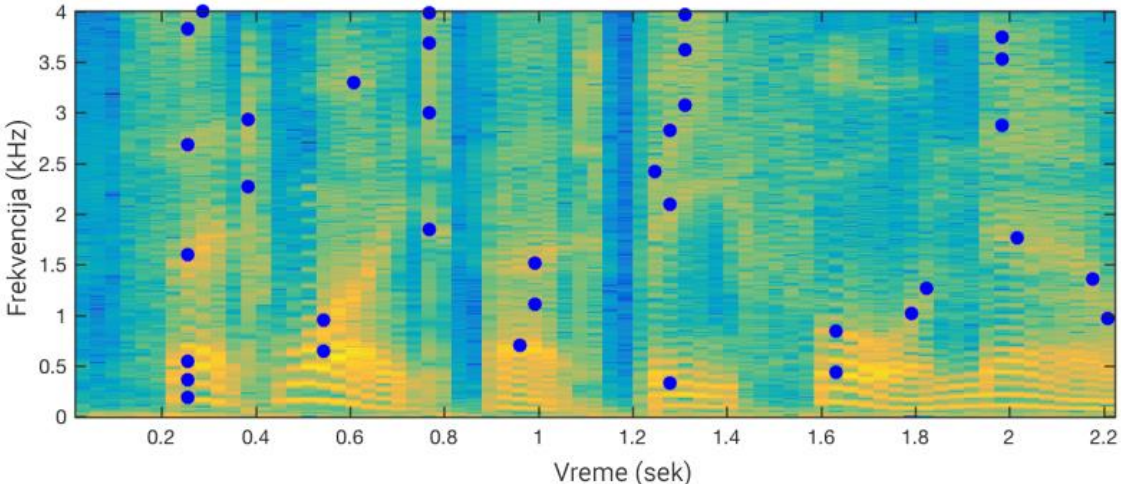
3.3 Ekstrakcija jedinstvenih tačaka iz audio signala

Ranije u poglavlju (3.2) „*Otisak prsta audio signala*“, predstavljena je polazna tačka za ekstrakciju važnih informacija iz audio signala. Uzimanje „otiska prsta“ u audio signalu je predproces za ekstrakciju jedinstvenih tačaka, jer kako je predstavljeno u (3.2), iako su dobijeni spektrogrami dva identična audio signala snimljena u dva različita okruženja (jedan bez pozadinskog šuma, drugi sa) potpuno različita, njihovi jedinstveni vrhovi-tačke su ostali netaknuti. Za identifikaciju i ekstrakciju vrhova danas postoje mnoge metode, jedna od njih je „*Gausova ekstrakcija vrhova*“ [14], ali ono ka čemu se stremi je izolacija delova spektrograma koji nisu povezani sa pozadinskim šumom, Slika 18.



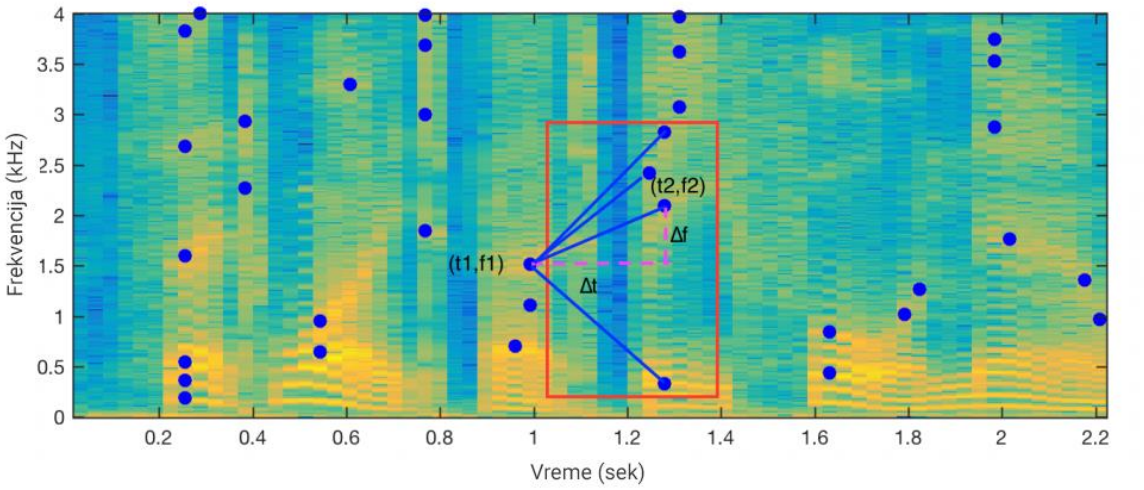
Slika 18. Gausova ekstrakcija vrhova

Identifikacija i ekstrakcija vrhova na audio spektrogramu upotrebom *Brze Furijerove Transformacije* (FFT) izgleda kao na Slika 19.



Slika 19. Extrakcija vrhova iz FFT spektrograma

3.4 Spektrogramsko filtriranje i kreiranje heš vrednosti



Slika 20. Kreiranje orjentira heš vrednosti

Identifikovani vrhovi u FFT spektrogramu se tretiraju kao sidrišne tačke oko koje se određuje vremensko ciljni prozor (crveni okvir na Slika 20), koji je u korelaciji sa određenim vrhom. Tako npr. tačka (t1, f1) se sekvencijalno uparuje sa N tačkama oko nje. Uparivanjem oko sidrišne tačke se dobija vektor koji je određen vremensko-frekventnom koordinatom sidrišne tačke i vremensko-frekventnom razlikom između tačaka u ciljnom prozoru, kao što se vidi na *Slika* 20. Tačka (t1, f1) i (t2, f2) se onda može predstaviti kao matematički izraz (11):

$$t1: [f_1, \Delta f, \Delta t], \Delta f = f_2 - f_1, \Delta t = t_2 - t_1 \quad (11)$$

- Izraz (11) se formuliše kao par: ***vremenski-pomeraj:heš***

3.4.1 Heširanje vrednosti

Nakon identifikacije i ekstrakcije jedinstvenih karakteristika audio signala, potrebno je samo još da se uskladišti u nekoj strukturi podataka oblika dobijenog u (11). Kreiranje heš tabele sidrišne tačke t_1: $[f_1, \Delta f, \Delta t]$ predloženog oblika kao u (11) bi izgledao:

$$ključ = f_1 \cdot 2^{N_{\Delta f}+N_{\Delta t}} + \Delta f \cdot 2^{N_{\Delta t}} + \Delta t \quad (12)$$

$$vrednost = ID \cdot 2^{N_{t1}} + t_1 \quad (13)$$

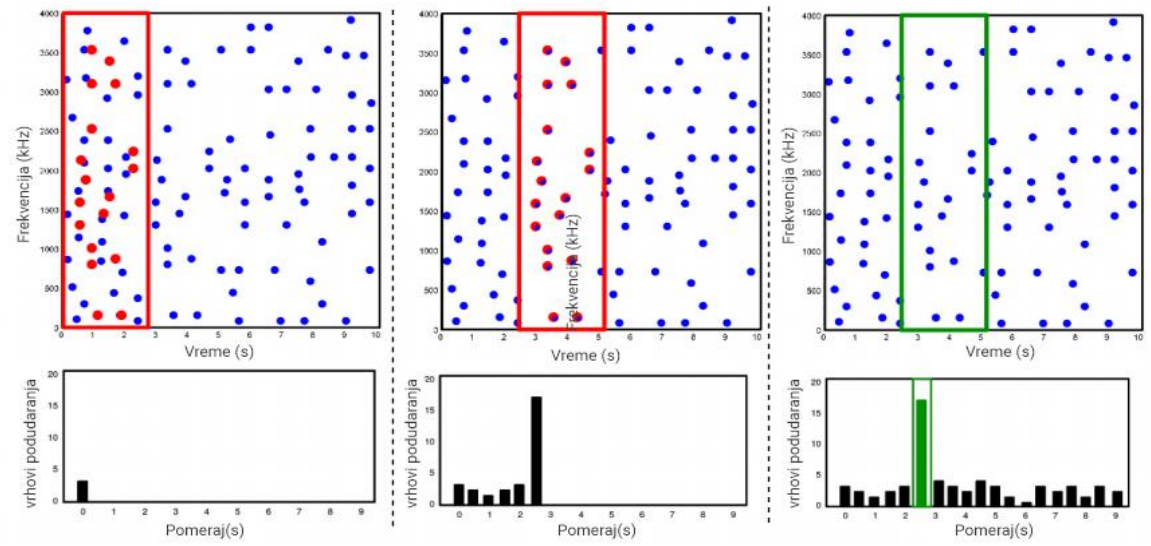
Gde su:

- $N_{t1}, N_{f1}, N_{\Delta t}$ *i* $N_{\Delta f}$ korišćene veličine u bitova da bi predstavili $t_1, f_1, \Delta t$ *i* Δf.
- ID je identifikacija audio fajla.

Ukupan broj različitih ključeva može da bude: $2^{N_{f1}+N_{\Delta f}+N_{\Delta t}}$.

3.5 Podudaranje

Proces pretraživanja audio materijala po sadržaju u svojoj srži se sastoji u nalaženju dva slična, ako ne i identična obrazca koji se nalaze u uskladištenoj heš vrednosti i heš vrednosti signala koji je pripremljen za pretragu. Audio signal koji je kandidat pretrage u većini slučajeva nije identičan indeksiranim-hešovanim audio bazama pretrage. Drugim rečima, snimljeni deo audio signala mora da prođe istu proceduru predprocesinga kao i indeksirani audio sadržaji koji se nalaze u bazi podataka. U toku snimanja signala koji je kandidat pretrage, obavezno se pojavljuje pozadinski šum, bilo da je to buka koja je u pozadini prilikom samog snimanja ili je to devijacija prilikom prenosa kroz kanal. Vremensko ciljni prozor „klizi“ po spektrogramu ekstraktovanih vrhova i upoređuje sa ciljnim prozorom kandidata za pretragu.



Slika 21. Podudaranje ciljnog prozora na spektrogramu vrhova [19]

Ciljni prozor klizi po spektrogramu i ukoliko dođe do najboljeg rangiranog prozora, podudaranje je najbolje, kao što može da se vidi na histogramima u Slika 21.

Iz prošlih nekoliko poglavlja zaključuje se da proces pretrage audio zapisa po sadržaju mora da prođe kroz čitav sistem pripreme, počevši od digitalizacije, kreiranja FFT spektrograma, ekstrakcije važnih vrhova iz FFT spektrograma, pa do određivanja vremensko ciljnog prozora i skladištenja heš vrednosti. Dobijene heš vrednosti se dijametralno razlikuju od kriptografskih heš vrednosti (MD5, SHA), koji u svojoj osnovi imaju da jedan bit promene rezultira promenu celog heš zapisa. U slučaju kreiranja heš zapisa audio FFT spektrograma i njegovo upoređivanje sa kandidatom pretrage, mora da bude otporno na devijaciju signala i iako signali koji se upoređuju nisu identični, podudaranje se nalazi.

4. Praktična realizacija aplikacije za pretraživanje muzičkih materijala

Eksperimentalna aplikacija za pretraživanje muzičkih materijala praktično je realizovana u programskom jeziku Java. [15]. Portabilnost, veliki broj uslužnih biblioteka i dobro poznavanje samog jezika, su razlog za izbor baš ovog jezika. Operativni sistem koji je korišćen prilikom razvoja aplikacije je Windows 10, koji je instaliran na desktop računaru konfiguracije:

- Procesor: Intel(R) Core(TM)2 Quad CPU Q8300 @ 2,50GHz 2,50GHz
- RAM memorija: 8GB
- Tip sistema: 64-bit Operativni sistem, 64-bit based processor
- Tip hard diska: SSD

Za potrebe razvoja aplikacije je preuzet i instaliran „*IntelliJ IDEA*“, integrisano razvojno okruženje (eng. ***I**ntegrated **D**evelopment **E**nvironment*). Razvila ga je kompanija pod imenom „*JetBrains*“ (nekada poznata kao [16]). Ovo razvojno okruženje je komercijalno, ali postoji i besplatna verzija koja se paralelno razvija sa komercijalnom [17].

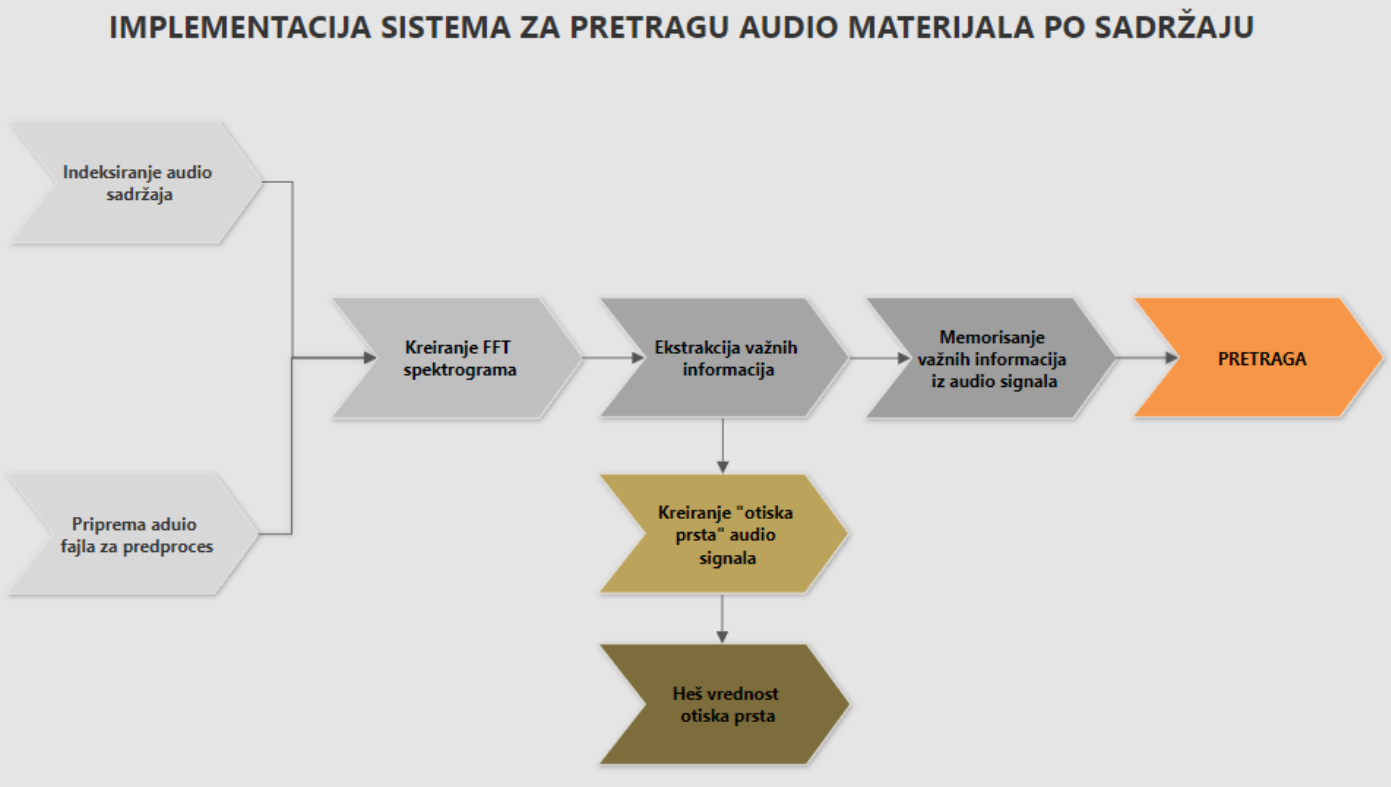
Pošto je sam projekat usmeren ka ekstrakciji važnih informacija iz audio sadržaja, koje su kasnije upotrebljene za pretragu, a ne za prikaz programerskih veština izrade, nije se koristila baza podataka, nego su podaci serijalizovani i sačuvani na tvrdi disk za kasniju upotrebu. Takođe je napravljen grafički korisnički interfejs (eng. ***G**raphical **U**ser **I**nterface* GUI) zbog lakšeg prezentovanja i korišćenja aplikacije.

Aplikacija je namenjena korisnicima koji žele da pretraže sve audio materijale na svom memorijskom medijumu po uzorku snimljenog sa mikrofona i odrede tačnu putanju do datoteka sa kojom se podudaraju. Ovo može biti korisno ukoliko imamo veliki broj audio sadržaja, a ne zna se njihova tačna lokacija na memorijskom medijumu. Takođe, može da se upotrebi i za pronalaženje dupliranih datoteka, pa samim tim ukoliko ih ima, brisanjem oslobodi prostor na računaru. Pre same pretrage potrebno je izvršiti par koraka koji su opisani kroz rad. Pa tako prvi korak koji je potrebno da se uradi je, digitalno učitavanje audio sadržaja sa diska za obradu, zatim kreiranje FFT spektrogramskog otiska audio datoteke, zatim ekstrakcija važnih informacija iz audio signala i kreiranje „Otisak prsta“ (eng. *Audio fingerprint*), nakon toga se kreira heš vrednost dobijenog „otiska prsta“ i na samom kraju, serijalizacija dobijenih heš vrednosti i čuvanje na tvrdi disk. Ovaj goreopisani postupak se radi pre pretrage sadržaja i možemo ga nazvati indeksiranje sadržaja za kasniju pretragu ili pretprocesorski deo pretrage.

- digitalno učitavanje audio sadržaja
- kreiranje FFT spektrogramskog otiska
- ekstrakcija važnih informacija iz audio signala
- kreiranje „otiska prsta“
- kreiranje heš vrednost dobijenog „otiska prsta“
- serijalizacija dobijenih heš vrednosti i čuvanje na tvrdi disk

Pošto je urađen pretprocesorski deo pretrage i hešovane vrednosti su serijalizovane na tvrdi disk, pristupa se samo pretrazi audio sadržaja. Prvi korak kod pretrage se sastoji u tome da se snimi audio sadržaj putem digitalnog uređaja i njega nazivamo uzorak-kandidat pretrage. Kada je uzorak snimljen, priprema za upoređivanje tako što se prebacuje u „*PCM_SIGNED 44100.0 Hz, 16 bit, mono, 4 bytes/frame, little-endian*“ format. Nakon toga se ponavljaju koraci kao u indeksiranju sadržaja i upoređuju se dobijene vrednosti sa serijalizovanom bazom, napravljenom u indeksnom procesu.

Predstava ovog sistema se može videti na Slika 22.



Slika 22. Dijagramski prikaz sistema za pretragu audio materijala po sadržaju

4.1 Digitalno učitavanje audio sadržaja

Ovaj proces je zajednički za oba subjekta u sistemu pretrage. Jedina razlika je u tome što se prilikom indeksiranja audio datoteke učitavaju u sistem sa memorijskog prostora na kome se nalaze, dok kod audio signala koji je predmet pretrage, podaci učitavaju sa mikrofona. Pošto je „MP3“ [18] najzastupljeniji format zapisa audio datoteke, a ujedno je i to zatvoren (eng. c*losed source*) format zaštićen patentom, upotrebljena je gotova Java biblioteka za učitavanje u sistem. Naziv projekta u kom je projektovana ova biblioteka je „*Soundlibs*“ [18].

4.2 Učitavanje audio datoteka za indeksiranje i pretragu

Slika 23. Prvi korak u implementaciji aplikacije

Prvi korak, Slika 23, u kom se pripremaju datoteke za indeksiranje je pretraga svih audio materijala na memorijskom medijumu za koji je korisnik zadao kriterijume. Kao što je napomenuto, ovaj proces je zajednički za oba subjekta sa malom razlikom da se u procesu indeksiranja iščitavaju datoteke sa memorijskog prostora na kom su. Navedeni kôd u Listing 1, se primenjuje za ovu operaciju :

```
public void ucitajDatotekeZaIndeksiranje(long idPesme, File datoteka)
        throws LineUnavailableException, IOException, UnsupportedAudioFileException {
    PCM2PCMConversionProvider provajderKonverzacije = new PCM2PCMConversionProvider();
    AudioInputStream in = AudioSystem.getAudioInputStream(file); //Binarno ucitavanje datoteke
    AudioFormat osnovniFormat = in.getFormat();
    System.out.println(osnovniFormat.toString());
    taskOutput.append(osnovniFormat.toString() + "\n"); //Azuriranje GUI interfejsa aplikacije
    AudioFormat dekodiranFormat = new AudioFormat(
        AudioFormat.Encoding.PCM_SIGNED,
        osnovniFormat.getSampleRate(), 16, osnovniFormat.getChannels(), //Postavljanje na format za FFT transformaciju
        osnovniFormat.getChannels() * 2, osnovniFormat.getSampleRate(),
        false); //Dekodiranje datoteke
    AudioInputStream din = AudioSystem.getAudioInputStream(dekodiranFormat, in);
    if (!provajderKonverzacije.isConversionSupported(getFormat(), dekodiranFormat)) //Provera da li je datoteka dekodirana
      System.out.println("Konverzija nije uspela !"); //Greska u konverziji
    System.out.println(dekodiranFormat.toString());
    AudioInputStream outDin = provajderKonverzacije.getAudioInputStream(getFormat(), din);
    AudioFormat formatTmp = dekodiranFormat;
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, formatTmp); //Otvaranje mikrofona
    TargetDataLine lineTmp = (TargetDataLine) AudioSystem.getLine(info);
    final TargetDataLine line = lineTmp;
    final AudioInputStream outDinSound = outDin;
    final long id = songId;
    Thread osnovniThread = new Thread(new Runnable() {

                                                    ...
```

Listing 1. Java kôd za učitavanje sadržaja audio datoteka za indeksiranje (Ceo kôd u dodatku Listing 1.)

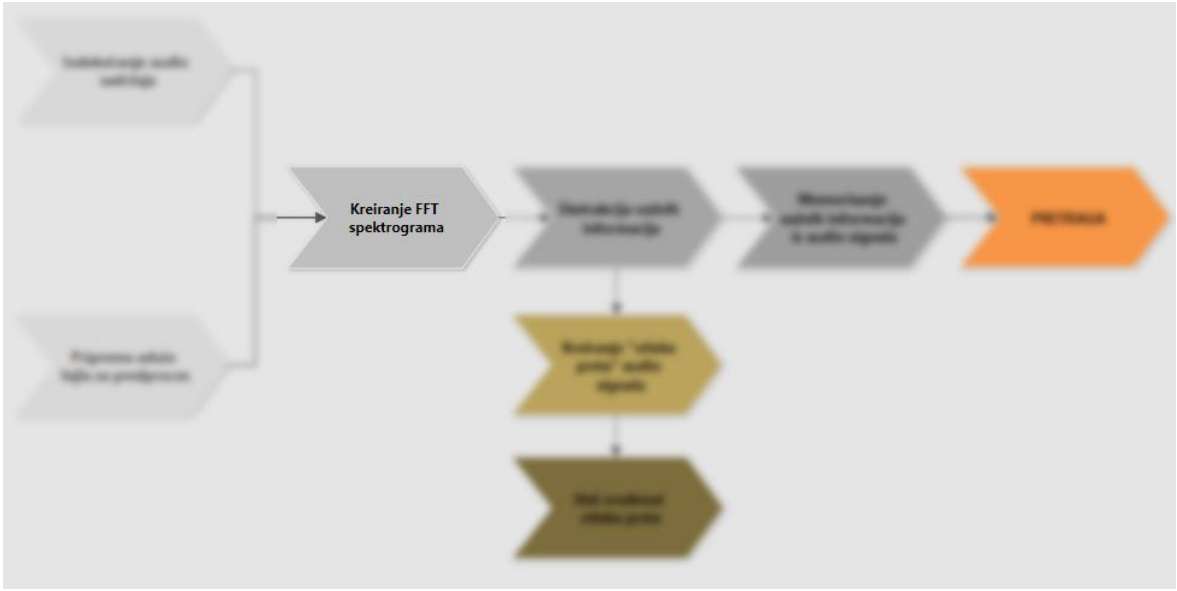
Kako je navedeno u Listing 1, audio za indeksiranje se učitava sa memorijskog medijuma, dok se datoteke za pretragu učitavaju sa mikrofona. Kôd u Listing 2, predstavlja taj proces.

```
public void ucitajDatotekuZaPretragu(long idPesme)
        throws LineUnavailableException, IOException, UnsupportedAudioFileException {
    new PCM2PCMConversionProvider();
    AudioFormat formatTmp = getFormat();
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, formatTmp);
    TargetDataLine lineTmp = (TargetDataLine) AudioSystem.getLine(info); //Otvaranje mikrofona
    final AudioFormat format = formatTmp;
    final TargetDataLine line = lineTmp;
    try {
       line.open(format);
       line.start();
    } catch (LineUnavailableException e) {
       e.printStackTrace();
    }
    final long id = idPesme;
    Thread osnovniThread = new Thread(new Runnable() {
       public void run() {
          ByteArrayOutputStream out = new ByteArrayOutputStream();

                                                    ...
```

Listing 2. Java kôd za učitavanje sadržaja audio datoteka za pretragu (Ceo kôd u dodatku Listinga 2.)

4.2.1 Kreiranje FFT spektrograma



Slika 24. Drugi korak u implementaciji aplikacije

Detaljan opis drugog koraka, Slika 24, dat je u poglavlju 3. „*Priprema digitalnog audio signala za pretraživanje*“, i u ovom koraku kreiramo FFT spektrogram da bi bili pripremljeni za sledeći korak, *ekstrakcije važnih informacija iz signala*.

Kôd u Listing 3 i Listing 4, daje pregled kako se pravi spektrogram audio signala.

```
private static Complex[] fft(Complex[] x) {
    int N = x.length;
    if (N == 1) return new Complex[]{x[0]};
    if (N % 2 != 0) throw new RuntimeException("N nije kvadrat od 2");
    Complex[] par = new Complex[N / 2];

                                                                    ...
```

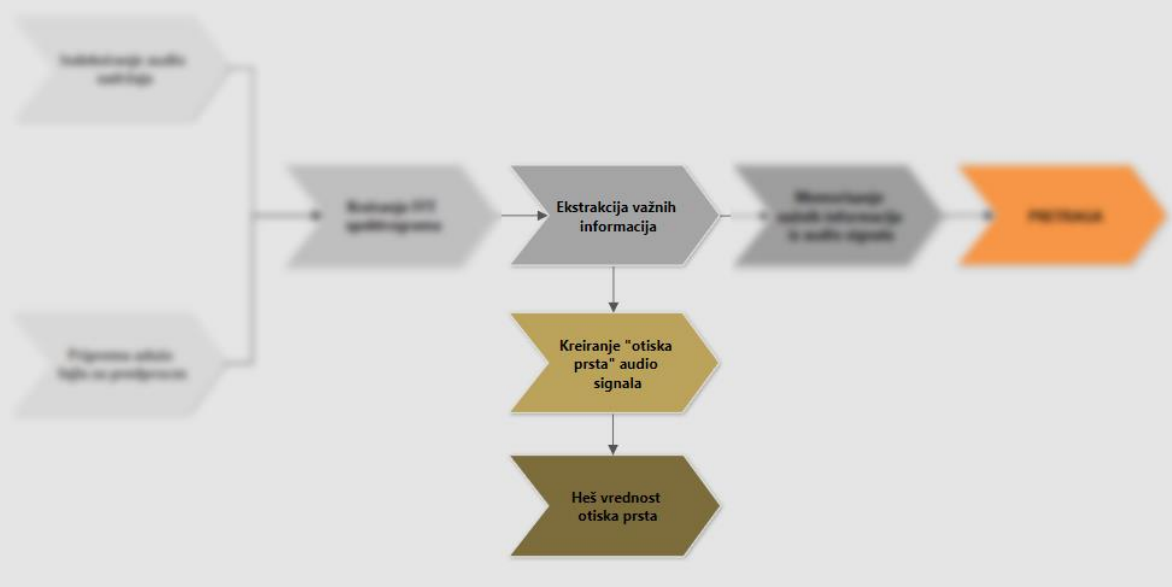
Listing 3. FFT transformacija audio signala (Ceo kôd u dodatku Listingu 3.)

```
public Complex[][] napraviSpectrum(ByteArrayOutputStream out) {
    byte audio[] = out.toByteArray();
    final int totalnaVelicina = audio.length;
    int moguceVrednosti = totalnaVelicina / 4096;

                                                                    ...
```

Listing 4. Dobijeni spektrum iz FFT transformacije signala (Ceo kôd u dodatku Listingu 4.)

4.2.2 Ekstrakcija važnih informacija iz audio signala



Slika 25. Treći korak u implementaciji aplikacije

Ekstrakcija najbitnijih i najvažnijih impulsa koji su potrebni za kreiranje heš vrednosti je detaljno opisana u poglavljima 3.2 *„Otisak prsta“ audio* signala i 3.3 „*Ekstrakcija jedinstvenih tačaka iz audio signala*“.

Ovaj korak se sastoji od dva pod koraka kao na Slika 25, a to su:

- Kreiranje „otiska prsta“ audio signala
- Kreiranje heš vrednosti dobijenog otiska prsta 3.4 „*Spektrogramsko filtriranje i kreiranje heš vrednosti*“

Programerska implementacija se vidi u Listing 5 i Listing 6, Java kôda.

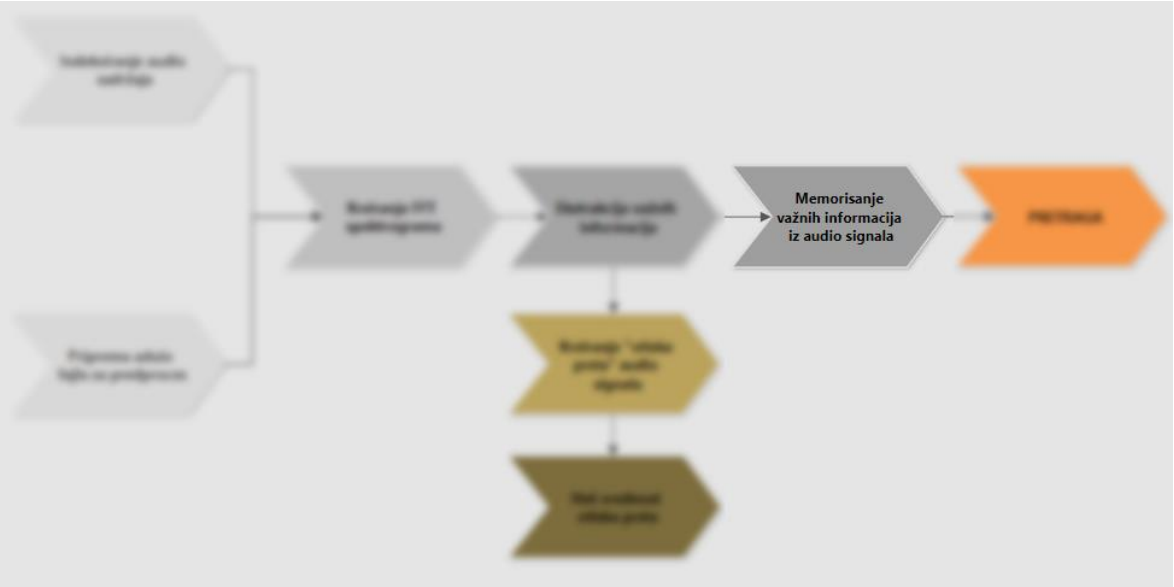
```
public void odrediKljučneTačke(Complex[][] rezultat, long idPesme, boolean daLiSePodudaraju) {
    this.mapaPodudaranja = new HashMap<Integer, Map<Integer, Integer>>();
    topMagnituda = new double[rezultat.length][5];
    for (int i = 0; i < rezultat.length; i++)
       for (int j = 0; j < 5; j++) topMagnituda[i][j] = 0;
    najvecaFrekvencija = new double[rezultat.length][GORNJI_LIMIT]; //Odredjivanje najvise frekvencije
    for (int i = 0; i < rezultat.length; i++)
       for (int j = 0; j < GORNJI_LIMIT; j++) najvecaFrekvencija[i][j] = 0;
    kljucneTacke = new long[rezultat.length][5];
    for (int i = 0; i < rezultat.length; i++)
       for (int j = 0; j < 5; j++) kljucneTacke[i][j] = 0;
    for (int t = 0; t < rezultat.length; t++) {
       for (int freq = DONJI_LIMIT; freq < GORNJI_LIMIT - 1; freq++) { //Odstranjivanje nepotrebnih frekvencija
          double mag = Math.log(rezultat[t][freq].abs() + 1);
          int index = getIndex(freq);
          if (mag > topMagnituda[t][index]) {
             topMagnituda[t][index] = mag;
             najvecaFrekvencija[t][freq] = 1;
             kljucneTacke[t][index] = freq; //Cuvanje kljucnih tacaka u HashMap
          }
       }
       long h = hash(kljucneTacke[t][0],
             kljucneTacke[t][1],
             kljucneTacke[t][2],
             kljucneTacke[t][3]);
       if (daLiSePodudaraju) { //Proces podudaranja
          List<DataPoint> listaTacaka;
          if ((listaTacaka = hashMap.get(h)) != null) {
             for (DataPoint dP : listaTacaka) {
                int ofset = Math.abs(dP.getTime() - t);
                Map<Integer, Integer> tmpMap = null;
                if ((tmpMap = this.mapaPodudaranja.get(dP.getSongId())) == null) {
                   tmpMap = new HashMap<Integer, Integer>();
                   tmpMap.put(ofset, 1);  //Popunjavanje HashMape sa dobijenim vrednostima
                   mapaPodudaranja.put(dP.getSongId(), tmpMap);
                   listaPesamaKojeSePodudaraju.add(dP.getSongId()); //Dodavanje najboljih kandidata podudaranja
                } else {
                   Integer broj = tmpMap.get(ofset);
                   if (broj == null) tmpMap.put(ofset, new Integer(1));
                   else tmpMap.put(ofset, new Integer(broj + 1));
                }
             }
          }
       } else {
          List<DataPoint> listaTacaka = null;
          if ((listaTacaka = hashMap.get(h)) == null) {
             listaTacaka = new ArrayList<DataPoint>();
             DataPoint tacka = new DataPoint((int) idPesme, t);
             listaTacaka.add(tacka);
             hashMap.put(h, listaTacaka);
          } else {
             DataPoint tacka = new DataPoint((int) idPesme, t); //Dodavanje kljucnih tacaka u listu
             listaTacaka.add(tacka);
          }
       }
    }
 }
```

Listing 5. Java kôd za ekstrakciju važnih tačaka u signalu ("Otisak prsta")

```
private long kreirajHash(long p1, long p2, long p3, long p4) { //Kreiranje jedinstvene hes vrednosti
    return (p4 - (p4 % FUZ_FACTOR)) * 100000000
          + (p3 - (p3 % FUZ_FACTOR)) * 100000
          + (p2 - (p2 % FUZ_FACTOR)) * 100
          + (p1 - (p1 % FUZ_FACTOR));
 }
```

Listing 6. Java kôd za kreiranje heš vrednosti ekstraktovanih tačaka

4.2.3 Memorisanje ekstraktovanih informacija iz audio signal



Slika 26. Četvrti korak: Memorisanje heš vrednosti iz "otiska prsta" audio signala

U ovom radu se pristupilo memorisanju heš vrednosti na tvrdi disk. Ovakav pristup je dobar zbog portabilnosti indeksiranih podataka. Ukoliko je ista struktura datoteka na drugim računarima, jednostavnim kopiranjem dobijenog indeksa (*.ser* datoteka) na drugi računar, moguće je korišćenje aplikacije bez ikakvih prepreka.

Prikazan kôd u Listing 7 i Listing 8, daje u uvid ovaj korak.

```
private void serijalizacija() {
      try {
         OutputStream fos =
               new FileOutputStream(putanjaDoIndeksiranihVrednosti+"Hes_Vrednosti_Muzike.ser"); //Kreiranje .ser datoteke
         ObjectOutputStream oos = new ObjectOutputStream(fos);
         oos.writeObject(hashMap);
         oos.close();
         fos.close();
         OutputStream fosL =

                                                                  ...
```

Listing 7. Java kôd za memorisanje heš vrednosti "otiska prsta"(Ceo kôd u dodatku Listingu 7.)

```
private void deserijalizacija() {
      try {
         FileInputStream fis = new FileInputStream(
               putanjaDoIndeksiranihVrednosti + "Hes_Vrednosti_Muzike.ser"); //Deserijalizacija hes vrednosti
         ObjectInputStream ois = new ObjectInputStream(fis);
         hashMap = (Map<Long, List<DataPoint>>) ois.readObject();
         ois.close();
         fis.close();
         FileInputStream fisL =
               new FileInputStream(putanjaDoIndeksiranihVrednosti + "Lista_Pesama.ser"); //Preuzimanje numera iz hes vrednosti

                                                                  ...
```

Listing 8. Java kôd za čitanje heš vrednosti "otiska prsta" (Ceo kôd u dodatku Listingu 8.)

4.2.4 Pretraga audio materijala po sadžaju

Slika 27. Peti korak: Pretraga audio materijala po sadržaju

U ovom koraku su sve potrebne informacije prikupljene. Informacije iz indeksiranih audio datoteka i audio predmeta pretrage, su serijalizovane i sačuvane na neku memorijsku lokaciju. Sada je samo potrebno da se uporedi heš vrednosti jedinstvenog „otiska prsta“ ova dva subjekta. Prikazan kôd u Listing 9, to i demonstrira.

```
private void pronadjiPodudaranjaPretrage() {
        int najboljePrebrojavanje = 0; //Inicijalno postavljanje parametra prebrojavanja na 0
        int najboljiKandidatPesme = -1;
        for (int id = 0; id < listaPesamaKojeSePodudaraju.size(); id++) {
            System.out.println("Za pesmu sa id-jem: " + listaPesamaKojeSePodudaraju.get(id));
            Map<Integer, Integer> tmpMap = mapaPodudaranja.get(listaPesamaKojeSePodudaraju.get(id));
            int najboljePrebrojavanjeZaPesmu = 0;
            for (Map.Entry<Integer, Integer> entry : tmpMap.entrySet()) {
                if (entry.getValue() > najboljePrebrojavanjeZaPesmu){
                    najboljePrebrojavanjeZaPesmu = entry.getValue();
                }
                System.out.println("Vremenski ofset = "
                        + entry.getKey()
                        + ", Broj = " + entry.getValue());  // Ispis vremneskog offseta u podudaranju
            }
            if (najboljePrebrojavanjeZaPesmu > najboljePrebrojavanje) {
                najboljePrebrojavanje = najboljePrebrojavanjeZaPesmu;
                najboljiKandidatPesme = listaPesamaKojeSePodudaraju.get(id) - 1;
            }
        }
        if (najboljiKandidatPesme < 0 || najboljePrebrojavanje < 2) {
            taskOutput.append("Nema pronadjene pesme \n");
            tag = "";
        } //Popunjavanje GUI interfejsa sa podacima
        taskOutput.append("Najbolja pesma je: \n");
        taskOutput.append("\t" + listaPesama.get(najboljiKandidatPesme).get(0) + "\n");
        taskOutput.append("Album: \n");
        taskOutput.append("\t" + listaPesama.get(najboljiKandidatPesme).get(1) + "\n");
        taskOutput.append("Izvodjac: \n");
        taskOutput.append("\t" + listaPesama.get(najboljiKandidatPesme).get(2) + "\n\n");
        System.out.println("Broj: " + najboljePrebrojavanje);
        listaPesamaKojeSePodudaraju.clear();
    }
```

Listing 9. Java kôd za podudaranje dve audio datoteke

5. Zaključna razmatranja

Cilj ovog diplomskog rada je da nakon detaljnog upoznavanja sa jedinstvenim karakteristikama audio signala, načinom na koji se one mogu "izvući", pa ujedno i kreiranje jedinstvenog "otisaka prsta" audio datoteke, omogući implementaciju aplikacije za pretragu audio materijala po sadržaju. Iako je aplikacija samo eksperimentalna, a dati vremenski rok za izradu veoma mali, rezultati su veoma ohrabrujući. Evidentni problemi spomenuti u radu, kao što su:

- brojevi audio datoteka
- memorijski prostori za skladištenje
- kompleksnost pri ekstrakciji važnih i jedinstvenih tačaka iz audio signala

su samo jedni od problema koji su morali biti rešeni u toku istraživanja i pisanja aplikacije. Jedinstvenim pristupom spektrogramske analize i upotrebe FFT transformacije za kreiranje heš vrednosti, problemi se svode na vremensko-prostorno prihvatljiv opseg za resurse "običnog" korisnika računara.

Iskustvo govori da je odgovor na najteže probleme u suštini „jednostavan", pa tako i naučni radovi [1], [11], koji su temelji multimilionskog biznisa, nude kreativno i jednostavno rešenje jako ozbiljnog problema u identifikaciji audio materijala. Ovim radom je na temeljima radova [1] i [11], i uz pomoć priloženih naučnih referenci, predloženo rešenje kako bi mogla da izgleda pretraga materijala po sadržaju.

Imajući u vidu vremenski "prozor" za izradu ovog rada, nije bilo mogućnosti za upotrebu dostupnih alata u domenu veštačke inteligencije. Kako je poznata činjenica da zadaci koji su laki ljudskom rodu, a teški za implementaciju na računarima, pravi kandidati za veštačku inteligenciju, predstavlja korak dalje u poboljšanju algoritama za identifikaciju i ekstrakciju važnih tačaka u audio signalima ovog rada. Najveća mana predloženog sistema je u tome, što kandidati za pretragu moraju da budu identični kao i materijali koji su indeksirani.

Primera radi, ako je audio datoteka određenog izvođača snimljenog u studijskom okruženju, indeksirana i spremna za upoređivanje, a kandidat pretrage snimljeni audio signal iste pesme i izvođača, samo snimljen na koncertu, mogućnost identifikacije-pretrage nije moguć za zadatu numeru, zbog prirode načina implementacije algoritma.

Ovo je problem koji je većini jedinkama ljudskog roda jako lako rešiv (po nekim eksperimentima za identifikaciju jako poznatih numera ide do 3 sekunde), dok je računarskim putem jako teško implementirati. Samim tim ovaj problem postaje jak kandidat za veštačku inteligenciju. Upotrebom predloženog rešenja bi se spomenuti nedostatak mogao svesti na osetno poboljšan i veći procenat podudaranja.

Literatura

[1] A. L.-C. Wang, *An Industrial-Strength Audio Search Algorithm,* Baltimore: Shazam Entertainment, Ltd, Oktobar 2003..

[2] M. Milosavljavić i S. Adamović, „Osnovi teorije informacija i kodovanja," u *Osnovi teorije informacija i kodovanja*, Beograd, Univerzitet Singidunum, 2013, pp. 12-13.

[3] D. d. V. Risojević, „OSOBINE I PERCEPCIJA ZVUKA," 2016. Dostupno na: http://dsp.etfbl.net/multimediji/2015/02%20GI%20Audio%20-%20percepcija%20i%20digitalizacija.pdf. [Poslednji pristup 22. 03. 2019.].

[4] „kluszeljka.weebly.com," Dostupno: https://kluszeljka.weebly.com/zvuk.html. [Poslednji pristup 20. 3. 2019.].

[5] Wikipedia, „Nyquist–Shannon sampling theorem," Dostupno na: https://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem. [Poslednji pristup 22. 04. 2019.].

[6] D. d. V. Risojević, „Uvod u digitalni audio," 15. 11. 2016. Dostupno na: http://dsp.etfbl.net/multimediji/2017/07_audio.pdf. [Poslednji pristup 27. 04. 2019.].

[7] Wikipedia.org, „Svetlost". Dostupno na: https://sr.wikipedia.org/wiki/Светлост. [Poslednji pristup 20. 04. 2019.].

[8] Shazam, „Shazam - Music Discovery, Charts & Song Lyrics". Dostupno na: https://www.shazam.com.

[9] S.Inc „SoundHound.Inc.". Dostupno na: https://www.soundhound.com.

[10] AcoustID, „Welcome to AcoustID". Dostupno na: https://acoustid.org/.

[11] T. K. Jaap Haitsma, „A Highly Robust Audio Fingerprinting System 2002". Dostupno na: http://ismir2002.ismir.net/proceedings/02-FP04-2.pdf. [Poslednji pristup 02. 05. 2019.].

[12] D. d. V. Risojević, „Spektralna analiza audio signala 2016".

[13] Wikipedia, „Brza furijeova transformacija". Dostupno na: https://sh.wikipedia.org/wiki/Brza_furijeova_transformacija. [Poslednji pristup 18. 05. 2019.].

[14] F. Liu, „Audio Fingerprinting for Speech Reconstruction and Recognition in Noisy," Dostupno: https://dspace.library.uvic.ca/bitstream/handle/1828/7912/Liu_Feng_MSc_2017.pdf. [Poslednji pristup 20. 05. 2019.].

[15] J. Oracle, „Oracle, Java“. Dostupno na: https://www.java.com/.

[16] D. T. f. Professionals, „Developer Tools for Professionals“. Dostupno na: https://www.jetbrains.com/.

[17] D. I. IDEA, „Jet Brains, Download IntelliJ IDEA: The Java IDE for Professional“ Dostupno na: https://www.jetbrains.com/idea/download/.

[18] Soundlibs„„https://github.com/pdudits/soundlibs“Dostupno: https://github.com/pdudits/soundlibs.

[19] M. Müller i J. Serrà, „Audio content-based music retrieval“. Dostupno na: http://ismir2011.ismir.net/tutorials/2011_MuellerSerra_MusicRetrieval_Tutorial-ISMIR_handouts-2.pdf. [Poslednji pristup 20. 05. 2019.].

Dodatak

Listing 1.

```
public void ucitajDatotekeZaIndeksiranje(long idPesme, File datoteka)
        throws LineUnavailableException, IOException, UnsupportedAudioFileException {
    PCM2PCMConversionProvider provajderKonverzacije = new PCM2PCMConversionProvider();
    AudioInputStream in = AudioSystem.getAudioInputStream(file);
    AudioFormat osnovniFormat = in.getFormat();
    System.out.println(osnovniFormat.toString()); //Ispis dobijenog osnovnog formata datoteke
    taskOutput.append(osnovniFormat.toString() + "\n");
    AudioFormat dekodiranFormat = new AudioFormat(
        AudioFormat.Encoding.PCM_SIGNED,
        osnovniFormat.getSampleRate(), 16, osnovniFormat.getChannels(),
        osnovniFormat.getChannels() * 2, osnovniFormat.getSampleRate(),
        false); // Postavljanje parametara za datoteku u pretprocesorskom delu
    AudioInputStream din = AudioSystem.getAudioInputStream(dekodiranFormat, in);
    if (!provajderKonverzacije.isConversionSupported(getFormat(), dekodiranFormat))
      System.out.println("Konverzacija nije uspela !");
    System.out.println(dekodiranFormat.toString());  //Ispis dekodirane datoteke
    AudioInputStream outDin = provajderKonverzacije.getAudioInputStream(getFormat(), din);
    AudioFormat formatTmp = dekodiranFormat;
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, formatTmp);
    TargetDataLine lineTmp = (TargetDataLine) AudioSystem.getLine(info); //Otvaranje mikrofona
    final TargetDataLine line = lineTmp;
    final AudioInputStream outDinSound = outDin;
    final long id = songId;
    Thread osnovniThread = new Thread(new Runnable() {
      public void run() {
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        running = true;
        int n = 0;
        byte[] buffer = new byte[(int) 1024];
        try {
          while (running) {
            n++;
            if (n > 1000) break;
            int count = 0;
            count = outDinSound.read(buffer, 0, 1024);
            if (count > 0) out.write(buffer, 0, count);
          }
          determineKeyPoints(napraviSpectrum(out), id, false); //Odredjivanje kljucnih tacaka
          out.close();
          line.close();
        } catch (IOException e) {
          System.err.println("I/O problem: " + e);
          System.exit(-1);
        }
      }
    });
    osnovniThread.start(); //Pokretanje osnovne niti
  }
```

Listing 1. Java kôd za učitavanje sadržaja audio datoteka za indeksiranje

Listing 2.

```
public void ucitajDatotekuZaPretragu(long idPesme)
        throws LineUnavailableException, IOException, UnsupportedAudioFileException {
    new PCM2PCMConversionProvider();
    AudioFormat formatTmp = getFormat();
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, formatTmp);
    TargetDataLine lineTmp = (TargetDataLine) AudioSystem.getLine(info);
    final AudioFormat format = formatTmp;
    final TargetDataLine line = lineTmp;
    try {
      line.open(format);
      line.start();
    } catch (LineUnavailableException e) {
      e.printStackTrace();
    }
    final long id = idPesme;
    Thread osnovniThread = new Thread(new Runnable() {
      public void run() {
        ByteArrayOutputStream out = new ByteArrayOutputStream(); //Citanje bitova datoteke
        radi = true;
        int n = 0;
        byte[] bafer = new byte[(int) 1024];
        Complex[][] rezultat; //Kreiranje COMPLEX broja za FFT
        try {
          while (radi) {
            n++;
            if (n % 40 == 0) taskOutput.append(Integer.toString(n / 40) + " sek\n");
            if (n > 1000) {  //25 seconds
              taskOutput.append("Završeno\n");
              System.out.println("Završeno");
              break;
            }
            int broj = 0;
            broj = line.read(bafer, 0, 1024);
            if (broj > 0) out.write(bafer, 0, broj);
          }
          try {
            rezultat = napraviSpectrum(out); //Pravljenje spektruma
            odrediKljučneTačke(rezultat, id, true);
          } catch (Exception e) {
            System.err.println("Greška: " + e.getMessage());
          }
          out.close();
          line.close();
        } catch (IOException e) {
          System.err.println("I/O problem: " + e);
          System.exit(-1);
        }
      }
    });
    osnovniThread.start();
  }
```

Listing 2. Java kôd za učitavanje sadržaja audio datoteka za pretragu

Listing 3.

```
private static Complex[] fft(Complex[] x) {
    int N = x.length;
    if (N == 1) return new Complex[]{x[0]};
    if (N % 2 != 0) throw new RuntimeException("N nije kvadrat od 2");
    Complex[] par = new Complex[N / 2];
    for (int k = 0; k < N / 2; k++) par[k] = x[2 * k];
    Complex[] q = fft(par);
    Complex[] nepar = par;
    for (int k = 0; k < N / 2; k++) nepar[k] = x[2 * k + 1];
    Complex[] r = fft(nepar);
    Complex[] y = new Complex[N];
    for (int k = 0; k < N / 2; k++) {
       double kth = -2 * k * Math.PI / N;
       Complex wk = new Complex(Math.cos(kth), Math.sin(kth)); //Kreiranje imaginarnog I realnog broja u Complex rezultatu
       y[k] = q[k].plus(wk.times(r[k]));
       y[k + N / 2] = q[k].minus(wk.times(r[k]));
    }
    return y;
}
```

Listing 3. FFT transformacija audio signala

Listing 4.

```
public Complex[][] napraviSpectrum(ByteArrayOutputStream out) {
    byte audio[] = out.toByteArray();
    final int totalnaVelicina = audio.length;
    int moguceVrednosti = totalnaVelicina / 4096;
    Complex[][] rezultat = new Complex[moguceVrednosti][];
    for (int vreme = 0; vreme < moguceVrednosti; vreme++) {
       Complex[] complex = new Complex[4096];
       for (int i = 0; i < 4096; i++) complex[i] = new Complex(audio[(vreme * 4096) + i], 0);
       rezultat[vreme] = fft(complex);
    }
    return rezultat;
}
```

Listing 4. Dobijeni spektrum iz FFT transformacije signala

Listing 5.

```
private void serijalizacija() {
      try {
         OutputStream fos =
               new FileOutputStream(putanjaDoIndeksiranihVrednosti+"Hes_Vrednosti_Muzike.ser");
         ObjectOutputStream oos = new ObjectOutputStream(fos);
         oos.writeObject(hashMap);
         oos.close();
         fos.close();
         OutputStream fosL =
               new FileOutputStream(putanjaDoIndeksiranihVrednosti+"Lista_Pesama.ser"); //Upis serijalizovanih vrednosti u datoteku
         ObjectOutputStream oosL = new ObjectOutputStream(fosL);
         oosL.writeObject(listaPesama);
         oosL.close();
         fosL.close();
      } catch (IOException ioe) {
         ioe.printStackTrace();
      }
   }
```

Listing 7. Java kôd za memorisanje heš vrednosti "otiska prsta"

Listing 6.

```
private void deserijalizacija() {
      try {
         FileInputStream fis = new FileInputStream(
               putanjaDoIndeksiranihVrednosti + "Hes_Vrednosti_Muzike.ser");
         ObjectInputStream ois = new ObjectInputStream(fis);
         hashMap = (Map<Long, List<DataPoint>>) ois.readObject();
         ois.close();
         fis.close();
         FileInputStream fisL =
               new FileInputStream(putanjaDoIndeksiranihVrednosti + "Lista_Pesama.ser"); //Ucitavanje serijalizovanih vrednosti
         ObjectInputStream oisL = new ObjectInputStream(fisL);
         listaPesama = (ArrayList<ArrayList<String>>) oisL.readObject();
         oisL.close();
         fisL.close();
      } catch (FileNotFoundException e) {
         System.out.println("'Hes_Vrednosti_Muzike.ser' & 'Lista_Pesama.ser' nije najena"); //Ispis da li je pronadjeno podudaranje

      } catch (IOException ioe) {
         System.out.println("Greska");
         ioe.printStackTrace();
         return;
      } catch (ClassNotFoundException c) {
         System.out.println("Klasa  nije nadjena");
         return;
      }

   }
```

Listing 8. Java kôd za čitanje heš vrednosti "otiska prsta"