

UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA RAČUNARSTVO I INFORMATIKU

SEMINARSKI RAD IZ PREDMETA MULTIMEDIJALNI SISTEMI

Konverzija RGB Slika u Zvučni Sadržaj

Studenti:

MUFTIĆ *Belma*, 1423/17260

KULOVIĆ *Nejra*, 1519/17484

KRUPALIJA *Ehlimana*, 1431/17461

Predmetni nastavnik:

r. prof. dr. HARIS ŠUPIĆ, dipl. ing. el.

Februar, 2019

Definicija Zadatka

U ovom radu opisan je postupak pretvaranja RGB slika u zvučni sadržaj. U ovu svrhu prvo je neophodno izdvojiti boje iz slike (budući da se rezultujuće zvučne datoteke razlikuju na osnovu različitog udjela boja u RGB slici), zatim je potrebno na adekvatan način dodijeliti određeni ton (od mogućih 128) nijansama boja, te kreirati zvučnu datoteku u *.mid* formatu. Kako bi se ovaj postupak demonstrirao, napravljena je *Windows Forms* aplikacija koristeći programski jezik *C#*, koja korisnicima omogućava odabir slike u boji za konverziju, pokretanje postupka konverzije te puštanje rezultujućeg zvučnog sadržaja.

U radu su detaljno opisane sve faze konverzije, uključujući razlaganje slike na *color channels*, mapiranje različitih boja u karakteristične frekvencije zvuka (tonove), odabir audio formata koji je najpogodniji za ovakvu konverziju te sam postupak kreiranja audio *file*-ova u odabranom formatu. Opisan je način implementacije aplikacije koja vrši konverziju, odnosno način na koji se slika razlaže na *color channels*, boje dodjeljuju frekvencijama, te date frekvencije koriste kako bi se kreirao audio *file* koji korisnici zatim mogu slušati.

Na kraju je opisan značaj ovakvog postupka, odnosno njegova moguća primjena u praksi za omogućavanje diferenciranja (i općenito identifikacije) različitih boja za ljude koji nisu u stanju da vide boje (ahromatopsija) ili razlikuju pojedine boje (daltonizam), te preciznost same konverzije.

Izjava o Doprinosu Članova Tima Pri Izradi Seminarskog Rada

Član tima	Aktivnosti
Belma Muftić	<p><i>Analiza problema:</i> Definicija teme</p> <p><i>Osmišljavanje rješenja:</i> Pronalazak rješenja za problem pretvaranja <i>color channels</i> u oblik pogodan za pretvaranje u zvučni sadržaj</p> <p><i>Praktična implementacija:</i> Dio rastavljanja slike na <i>color channels</i> i pretvaranje u ekvivalentne frekvencije zvuka</p> <p><i>Pisanje teksta:</i> Za prethodno navedeni dio problema</p>
Nejra Kulović	<p><i>Analiza problema:</i> Odabir tehnologije za razvoj rješenja</p> <p><i>Osmišljavanje rješenja:</i> Ideja za rješavanje problema dodjeljivanja odgovarajućih frekvencija zvuka dekodiranim bojama</p> <p><i>Praktična implementacija:</i> Aplikacija koja objedinjuje zasebne dijelove za manipulisanje slikom i zvukom te njihovo spajanje i usklađivanje</p> <p><i>Pisanje teksta:</i> Za prethodno navedeni dio problema</p>
Ehlimana Krupalija	<p><i>Analiza problema:</i> Odabir vrste protokola za rezultujuće zvučne <i>file</i>-ove</p> <p><i>Osmišljavanje rješenja:</i> Prilagodavanje <i>.mid</i> formata primjeni te pretvaranje dodijeljenih frekvencija u audio format</p> <p><i>Praktična implementacija:</i> Dio definisanja dijelova MIDI <i>file</i>-a, ubacivanja zvučnog sadržaja i kreiranja <i>.mid file</i>-a</p> <p><i>Pisanje teksta:</i> Za prethodno navedeni dio problema</p>

Tabela 1: Opis aktivnosti članova tima pri rješavanju problema

Sadržaj

1	Uvod	1
1.1	Postavka problema	1
1.2	Rastavljanje RGB slika na <i>color channels</i>	1
1.3	Mapiranje pojedinih boja u frekvencije zvuka	1
1.4	Kreiranje zvučnog sadržaja na osnovu boja	1
1.4.1	Odabir audio formata	1
1.4.2	<i>Musical Instrument Digital Interface</i> format	2
2	Implementacija rješenja problema	6
2.1	Manipulacija slikom	6
2.2	Manipulacija zvukom	6
2.3	Aplikacija za pretvaranje slika u boji u zvučni sadržaj	8
3	Zaključak	9

1 Uvod

1.1 Postavka problema

Problem koji će biti riješen u okviru ovog rada je pronalazak načina da se premosti prepreka koju predstavlja nemogućnost viđenja ili razlikovanja boja. Budući da osobe koji nisu u stanju razlikovati ili vidjeti boje vide slike u boji kao da su *grayscale*, neophodno je izvršiti ekstrakciju onih elemenata koje oni ne vide - boje - te njihovu konverziju u sadržaj koji će moći razlikovati, što je u najvećem broju slučajeva zvuk. Iz tog razloga izvršiti će se postupak konverzije RGB slika u zvučni sadržaj.

U ovu svrhu potrebno je definisati tri glavne faze konverzije:

1. Izdvajanje boja iz slike, odnosno adekvatno razlaganje slike na određeni broj nijansi;
2. Mapiranje pojedinih boja u frekvencije zvuka;
3. Kreiranje audio *file*-a na osnovu informacija o tonovima od kojih se ista treba sastojati.

U nastavku će biti detaljno objašnjene sve faze konverzije, odnosno način ekstrakcije boja iz slike, njihovog mapiranja u različite frekvencije zvuka (koje će predstavljati način razlikovanja boja) te spajanja različitih boja u jedinstven audio *file* koji će zatim biti moguće reproducirati te na taj način putem slušnog organa omogućiti identifikaciju različitih boja.

1.2 Rastavljanje RGB slika na *color channels*

% Belmin dio

1.3 Mapiranje pojedinih boja u frekvencije zvuka

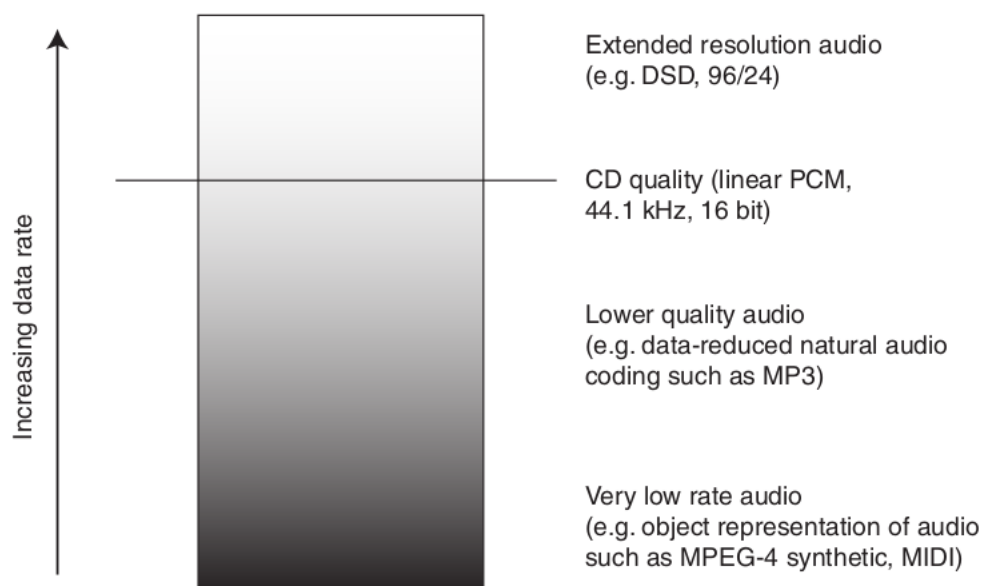
% Nejrin dio

1.4 Kreiranje zvučnog sadržaja na osnovu boja

1.4.1 Odabir audio formata

Postoji veliki broj audio formata koji omogućavaju čuvanje zvučnih sadržaja nastalih prirodnim ili vještačkim putem, no većina takvih formata zahtijeva **uzorkovanje signala** te veliku preciznost kako bi izlazni *file*-ovi bili dovoljno kvalitetni. S povećanjem broja uzoraka u jedinici vremena povećava se kvalitet zvučnog sadržaja, no u isto vrijeme, povećava se i **memorija potrebna za njihovo čuvanje**.

Prema Nyquistovom teoremu o uzorkovanju, frekvencija uzorkovanja signala mora biti barem dva puta veća od kritične frekvencije prisutne u izvornom signalu kako bi se signal mogao uspješno rekonstruisati. U slučaju zvuka, ljudski slušni sistem ima najvišu frekvenciju mehaničke rezonance u iznosu od **20 kHz**, te iz tog razloga frekvencija uzorkovanja treba biti **barem 40 kHz** kako ne bi došlo do pojave *aliasinga* (pogrešne rekonstrukcije signala zbog nedovoljnog uzorkovanja) [1]. Na Slici 1 prikazan je trend porasta zauzeća memorije pri porastu kvaliteta prirodnih audio sadržaja, te je vidljivo da veći broj audio formata nema dovoljnu frekvenciju uzorkovanja za rekonstrukciju ulaznih zvučnih signala. [2]



Slika 1: Odnos količine podataka s frekvencijom uzorkovanja audio signala [2]

U slučaju pretvaranja slika u zvučni sadržaj, nema razloga za korištenje audio formata koji imaju visoku frekvenciju uzorkovanja, odnosno koji su prilagođeni za uzorkovanje ulaznih signala (poput RIFF, PCM ili DSD formata), budući da je manja frekvencija uzorkovanja, tj. manja preciznost pri kvantizaciji, **sasvim dovoljna** kako bi vještački kreiran zvuk bio dovoljno kvalitetan. Jedan od formata koji je pogodan za ovakvu obradu je **MIDI format**, o kojem će biti više riječi u nastavku.

1.4.2 *Musical Instrument Digital Interface format*

MIDI predstavlja protokol kreiran ranih 1980-ih godina kako bi se omogućilo **kontrolisanje muzičkih instrumenata** i svih njihovih aspekata (visina, jačina tona, pritisak na tipku, držanje tipke, ritam, sinkope i sl.), što nijedan prethodni format nije razmatrao. Glavna razlika između standardnih formata za digitalizaciju zvuka i MIDI formata je u tome što se u ovom formatu **ne čuvaju podaci o uzorcima zvučnog signala**, tako da se omogućava kreiranje *file*-ova koji se ne sastoje od velikog broja uzoraka frekvencija zvuka u vremenu.

U ovom formatu, zvuk poprima jednu od **128 karakterističnih frekvencija** (odnosno, frekvencija predstavljenih 16-bitnim vrijednostima) odabranih za predstavljanje muzičkih nota (pri čemu je najniži ton C₀ kodiran heksadecimalnom vrijednošću 00 00, te ima frekvenciju od 16.35 Hz, a najviši ton B₉ kodiran je heksadecimalnom vrijednošću FF FF i ima frekvenciju 15,804.26 Hz [3]). Na ovaj način postiže se da je veličina *file*-ova koji koriste ovaj protokol **mala u odnosu na druge formate**, jer nema potrebe za čuvanjem velikog broja uzoraka po jedinici vremena, zbog čega je za posljedicu i veličina audio *file*-ova koji koriste MIDI format manja u odnosu na druge audio formate. [2]

Posljedica mapiranja svih frekvencija zvuka u 128 karakterističnih frekvencija je **manji kvalitet zvuka**, no na ovaj način omogućava se kreiranje **potpuno vještačkog signala** (čemu je ovaj format primarno i namijenjen), što je veoma teško ili nemoguće izvršiti koristeći druge formate, budući da to nije njihova primarna namjena (već digitalizacija prirodnog zvučnog signala te što vjernije predstavljanje istog).

Još jedna važna razlika između MIDI formata i standardnih formata je u tome što je zvučni signal u standardnim formatima potpuno definisan, te je kao takav **na svim izlaznim uređajima uvijek isti**. MIDI format definiše veliki broj različitih aspekata važnih za izlazni signal i njegove pojedine dijelove, no sam ton (odnosno, karakteristična frekvencija) predstavlja samo **smjernicu** za konačni izlazni signal, što omogućava da se za isti ulazni *file* dobiju različiti izlazni signali koristeći različite izlazne uređaje. Budući da su izlazni uređaji najčešće simulatori muzičkih instrumenata, na ovaj način postiže se jednostavna tranzicija između njih, te nije potrebno uvoditi dodatne specifikacije kako bi se omogućio ovaj efekat. Ovakvo nešto nemoguće je izvršiti koristeći standardne audio formate. [2]

U Tabeli 2 prikazana je struktura kompletnog MIDI *file*-a, o kojoj će biti riječi u nastavku.

<i>MIDI Header</i>	<i>Track Header</i>	<i>Track Data</i>
4D 54 68 64 00 00 00 06 00 XX D E	4D 54 72 6B G	[...] 00 FF 2F 00

Tabela 2: Struktura MIDI *file*-a

File-ovi u MIDI formatu sastoje se od sljedećih dijelova [4]:

1. **MIDI Header**, koji specificira format *file*-a:

- **Sekcija A:** Označava da je format *file*-a MIDI, te se sastoji od 4 *byte*-a koji imaju heksadecimalnu vrijednost 4D 54 68 64;
- **Sekcija B:** Specificira veličinu ostatka MIDI *header*-a, odnosno broja *byte*-a do početka informacija o samom signalu, te se sastoji od 4 *byte*-a koji imaju heksadecimalnu vrijednost 00 00 00 06 (budući da ostali dijelovi *header*-a imaju fiksnu dužinu);
- **Sekcija C:** Označava tip MIDI formata. Postoje tri moguća tipa: *tip 0* (osnovni tip), *tip 1* (omogućava korištenje do 2^{16} različitih *channels* za definisanje različitih signala koji se pokreću u isto vrijeme) i *tip 2* (koji predstavlja više MIDI *file*-ova u jednom kako bi se definisali različiti šabloni zvuka). Ova sekcija sastoji se od 2 *byte*-a (za najčešće korišteni tip 1, heksadecimalna vrijednost ove sekcije je 00 01);
- **Sekcija D:** Označava broj *channels* koji će se koristiti u *file*-u, te se sastoji od 2 *byte*-a;
- **Sekcija E:** Označava brzinu signala, odnosno ritam zvuka, te se sastoji od 2 *byte*-a.

U Tabeli 3 prikazana je struktura ovog dijela MIDI *file*-ova.

A	B	C	D	E
4D 54 68 64	00 00 00 06	00 XX	YY YY	ZZ ZZ

Tabela 3: Struktura MIDI *header*-a

2. **Track Header**, koji daje osnovne informacije o samom zvučnom signalu:

- **Sekcija F:** Označava početak podataka o zvučnom signalu, te se sastoji od 4 *byte*-a koji imaju heksadecimalnu vrijednost 4D 54 72 6B;

- **Sekcija G:** Specificira broj *byte*-a u ostatku *file*-a, odnosno veličinu samog zvučnog signala. Budući da se ova sekcija sastoji od 4 *byte*-a, najveća moguća veličina zvučnog signala je 2^{32} *byte*-a.

U Tabeli 4 prikazana je struktura ovog dijela MIDI *file*-ova.

F				G			
4D	54	72	6B	XX	XX	XX	XX

Tabela 4: Struktura *track header*-a

3. **Track Data**, koji sadrži sam signal, odnosno njegove pojedine dijelove (muzičke tonove). Svaki ton sastoji se iz sljedećih dijelova:

- **Timestamp:** Označava broj vremenskih jedinica koje trebaju proći prije početka sljedećeg tona, odnosno njegovo kašnjenje. Za čekanja do 127 otkucaja (7F) ova sekcija sastoji se od jednog *byte*-a.

Za sva veća čekanja, dodaje se po jedan *byte* vrijednosti 81 koja se zatim povećava po potrebi (naprimjer, čekanje od 128 otkucaja specificira se kao 81 7F, pri čemu se ne uvodi novi *byte* dok se ne dostigne vrijednost FF 7F i sl.).

Razlog za ovakvo kodiranje je što je potrebno biti moguće kodirati velika kašnjenja, a u isto vrijeme treba se moći identificirati mjesto na kojem informacija o kašnjenju završava, što ne bi bilo moguće u slučaju da svi *byte*-i mogu poprimiti vrijednosti od 00 do FF;

- **Status:** Gornja četiri bita ovog *byte*-a označavaju događaj (*event*), odnosno akciju koju je potrebno izvršiti prije početka sljedećeg tona (moguće vrijednosti prikazane su u tabeli 6), dok donja četiri bita označavaju *channel* na koji se sljedeći ton i sve specificirane postavke odnose.
- **Pitch:** Predstavlja karakterističnu frekvenciju, odnosno sljedeći ton u zvučnom signalu, te se sastoji od jednog *byte*-a (vrijednosti od 00 do 7F);
- **Volume:** Predstavlja jačinu tona, te se sastoji od jednog *byte*-a (vrijednosti od 00 do 7F).

U Tabeli 5 prikazana je struktura ovog dijela MIDI *file*-ova.

Timestamp		Status		Pitch	Volume
[...]	XX	Y	Z	AA	BB

Tabela 5: Struktura pojedinih tonova

Statusni bit	Značenje
8	Ugasi ton
9	Upali ton
A	Pritisak tipke instrumenta (<i>AfterTouch</i>)
B	Kontroler (simulacija fizičkih osobina uređaja) [2]
C	Promjena programa (simulacija efekata) [2]
D	Jačina pritiska na tipku
E	Visina tona

Tabela 6: Moguće vrijednosti statusnih bita

Važno je napomenuti da se nakon posljednjeg tona mora definisati kraj zvučnog sadržaja ubacivanjem vrijednosti `00 FF 2F 00`. Ovo je jedna od formi *meta events*, odnosno lažnih događaja koji se kodiraju na isti način kao i stvarni događaji koji proizvode tonove, no kao posljedicu nemaju novi ton, već neku drugu akciju. Svi lažni događaji sastoje se iz četiri dijela:

- Prvi *byte* označava da je u pitanju *meta event* i uvijek ima heksadecimalnu vrijednost `FF`;
- Drugi *byte* specificira vrstu *meta event*-a (informacije o samom *file*-u (broj snimka, autor, godina, ime kompozicije i sl.), tempo, MIDI port, itd.) [2];
- Treći dio specificira broj *byte*-a od kojih se sami podaci sastoje;
- Četvrti dio sadrži same podatke.

U Tabeli 7 prikazana je struktura ovog dijela MIDI *file*-ova.

1	2	3	4
FF	XX	YY	[...]

Tabela 7: Struktura *meta event*-a

2 Implementacija rješenja problema

2.1 Manipulacija slikom

% Belmin dio

2.2 Manipulacija zvukom

Treća faza konverzije RGB slika u zvuk je **kreiranje MIDI *file*-a** s tonovima koji predstavljaju odgovarajuće boje od kojih se slika sastoji. Kako bi se to omogućilo, kreirana je klasa `MIDIFile`, čiji atributi predstavljaju pojedine dijelove MIDI *file*-a koji se sukcesivno kreira, odnosno koji ima predefinisane dijelove (*header* dijelove), a dio koji se odnosi na same podatke (tj. tonove) inicijalno je prazan. Atributi klase `MIDIFile` prikazani su u Listingu 1.

Listing 1: Klasa `MIDIFile` i njeni atributi

```
1 public class MIDIFile
2 {
3     #region MIDIHeader
4     string format = "4D546864"; // specifies that the file format is
        MIDI (MThd)
5     string headerSize = "00000006"; // specifies the number of bytes of
        // the following three parts of the MIDI header (SMF)
6     string type = "0001"; // specifies the type of MIDI (0, 1 or 2)
7     string tracks = "0001"; // specifies the number of tracks (0 -
        65.536)
8     string speed = "0080"; // specifies the speed of music
9     #endregion
10    #region TrackHeader
11    string start = "4D54726B"; // specifies the track start (MTrk)
12    string noOfBytes = ""; // specifies the number of bytes in the track
13    #endregion
14    #region Track
15    string data = ""; // specifies the musical notes in the track
16    string rhythm = "8100"; // specifies the speed of each note
17    string volume = "60"; // specifies the volume of each note
18    string end = "00FF2F00"; // specifies the track end
19    #endregion
20 }
21
```

Klasa `MIDIFile` omogućava **dodavanje novog tona** na vrlo jednostavan način. Kao parametar se prima cjelobrojna vrijednost između 0 i 127 (koja predstavlja karakterističnu frekvenciju tona), te se zatim provjerava da li je u pitanju prvi ton ili ne. Ukoliko je u pitanju prvi ton, potrebno je poslati signal da se pritisne prva tipka, dok je u svakom drugom slučaju dovoljno dodati samo broj otkucaja koji se čekaju do početka sljedećeg tona. Zatim se dodaje vrijednost koja označava karakterističnu frekvenciju (uz dodatnu provjeru da li karakteristična vrijednost ima dva *byte*-a, te ukoliko nema, drugi *byte* se manuelno dodaje), te trenutno postavljena vrijednost jačine zvuka. Ova funkcionalnost implementirana je u funkciji `addNote`, čija je struktura prikazana u Listingu 2.

Listing 2: Metoda za dodavanje tonova u MIDI *file*

```
1 // adds a new note (range from 0 to 127) to the track
```

```

2      public void addNote(int note)
3      {
4          if (data.Length == 0) data += "800090"; // the first note needs
5              // to send the signal to turn the notes on
6          else data += rhythm;
7          if (note < 16) data += "0";
8          data += note.ToString("X") + volume;
9      }

```

Nakon dodavanja svih željenih tonova, potrebno je kreirati *.mid file* koji se sastoji od svih prethodno navedenih dijelova. Nakon dodavanja posljednjeg, "tihog" tona koji signalizira otpuštanje pritiska na tipku, svi dijelovi (*headers* i *data*) sastavljaju se u **jedinstvenu varijablu** koja se sastoji od heksadecimalnih vrijednosti prikazanih u obliku karaktera. Ti karakteri zatim se pretvaraju u binarne vrijednosti (2 po 2 karaktera, kako bi se dobila 16-bitna vrijednost) koje se redom upisuju u **binarni file s ekstenzijom .mid**. Ova funkcionalnost implementirana je u funkciji `createMIDIFile`, čija je struktura prikazana u Listingu 3.

Listing 3: Spajanje svih dijelova MIDI *file-a* i kreiranje *.mid file-a*

```

1      // puts all file parts together and writes to .mid file
2      public void createMIDIFile(string path)
3      {
4          data += rhythm + "B07B00"; // the final note just turns off the
5              music
6          noOfBytes = ((data.Length + end.Length) / 2).ToString("X");
7              // previously skipped part of TrackHeader
8          while (noOfBytes.Length < 8) noOfBytes = "0" + noOfBytes;
9              // the noOfBytes part needs to have 8 bytes
10         string file = format + headerSize + type + tracks + speed +
11             start + noOfBytes + data + end; // putting all
12             parts together
13         // writing to .mid file - hexadecimal code is converted to
14             binary and then written to file
15         var stream = new FileStream(path + ".mid", FileMode.Create,
16             FileAccess.ReadWrite);
17         var twoCharacters = new StringBuilder(); // two bytes are used
18             for every conversion (16 bits - two hex numbers)
19         var singleByte = new byte[1]; // two binary bytes to which the
20             hex numbers will be converted
21         foreach (var character in file)
22         {
23             twoCharacters.Append(character); // adding one hex character
24                 to the 16-bit variable
25             if (twoCharacters.Length == 2) // added two characters -
26                 reached 16 bits
27             {
28                 singleByte[0] = (byte)Convert.ToByte(twoCharacters.
29                     ToString(), 16); // conversion from hex to bin
30                 stream.Write(singleByte, 0, 1); // writing bin to file
31                 twoCharacters.Clear(); // starting over again with new
32                     characters
33             }
34         }
35         stream.Close();
36     }

```

2.3 Aplikacija za pretvaranje slika u boji u zvučni sadržaj

% Nejrin dio

3 Zaključak

Razvoj tehnologije omogućio je jednostavnu i brzu manipulaciju multimedijalnim sadržajima. Postoji veliki broj različitih formata slika te gotovih programskih rješenja i biblioteka koje omogućavaju njihovu analizu, izmjenu ili ekstrakciju onih dijelova koji su važni za određenu primjenu. S druge strane, iako je obrada zvukom manje popularna, postoji veliki broj audio formata koji omogućavaju tretiranje zvuka na različite načine, kompresiju ili kreiranje potpuno vještačkih zvučnih signala, što također može biti korisno i od velike pomoći za različite primjene. Najvažnije je omogućavanje izvršavanja svih ovih obrada **u realnom vremenu**, što otvara vrata velikom broju mogućnosti.

Jedna od takvih mogućnosti je učitavanje slike, njeno razlaganje na *color channels* te njihovo pretvaranje u audio *file* koji je moguće reproducirati gotovo odmah. Na ovaj način omogućava se da boje, koje neki ljudi nisu u stanju vidjeti, budu pretvorene u karakterističnu frekvenciju koja, na isti način kao što oko prepoznaje karakteristične boje, bude prepoznata od strane slušnog sistema. Na ovaj način moguće je ukloniti prepreku koju predstavlja **nemogućnost viđenja boja**, odnosno omogućiti ljudima koji nisu u stanju da razlikuju boje (daltonizam) ili koji vide samo nijanse sive boje (ahromatopsija) da **”čuju” boje**, odnosno da razlikuju boje na osnovu različite frekvencije zvuka koja se proizvodi na osnovu mapiranja boja u muzičke tonove.

Naravno, ovakav sistem nije savršen, jer postoji na milione različitih nijansi koje je ljudsko oko u stanju razlikovati, dok postoji samo 128 različitih muzičkih tonova, čime se većina nijansi zapravo **aproksimira** u jednu od karakterističnih frekvencija. Ovo je posljedica činjenice da vidni i čulni sistemi ne funkcionišu na isti način: u bazelarnoj membrani dolazi do pojave **frekvencijskog maskiranja**, pri čemu se bliske frekvencije detektuju kao jedna, što predstavlja jednu vrstu aproksimacije [1]. Iz ovog razloga i postoji samo 128 različitih muzičkih tonova, zbog čega se nameće zaključak da je vršenje aproksimacija **neizbježno** pri mapiranju boja u frekvencije zvuka.

Iako je potpuno preslikavanje skupa boja u skup zvukova nemoguće, preslikavanje 128 boja je ipak dovoljno detaljan postupak kako bi se stvorio osjećaj razlikovanja boja. Na ovaj način omogućava se pojava **sinestezije**, odnosno nakon određenog vremena korištenja sistema za pretvaranje slika u boji u zvukove, ljudski čulni sistem koji nije u mogućnosti razlikovati boje početi će vršiti automatsku identifikaciju boja koje ne vidi na osnovu zvuka koji čuje. Omogućavanje viđenja boja i u slučaju kada je čulni organ oštećen ili defektan, bez obzira s kolikom aproksimacijom, veliki je uspjeh, jer može predstavljati veliku pomoć ljudima koji nisu u stanju razlikovati ili vidjeti boje, a za realizaciju ovakvog sistema nisu potrebne velike sume novca niti komplikovani postupci.

Reference

- [1] H. Šupić, *Kompresija Podataka*. Sarajevo, Bosnia and Herzegovina: Univerzitet u Sarajevu, 2016.
- [2] F. Rumsey, *Desktop Audio Technology: Digital Audio and MIDI Principles*, ser. Music Technology Series. Oxford, United Kingdom: Focal Press, 2004, vol. IV.
- [3] (2019, February) Table of musical notes and their frequencies and wavelengths. [Online]. Available: <https://www.liutaiomottola.com/formulae/freqtab.htm>
- [4] (2019, February) A crash course on the standard midi specification. [Online]. Available: <http://www.skytopia.com/project/articles/midi.html>