Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Generovanie farebných notových zápisov pre xylofón

Bakalárska práca

2023
Lenka Ráčková

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Generovanie farebných notových zápisov pre xylofón

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Informatika
Školiace pracovisko:	FMFI.KAI - Katedra aplikovanej informatiky
Školiteľ:	RNDr. Zuzana Berger Haladová, PhD.

Bratislava, 2023
Lenka Ráčková

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Lenka Ráčková
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Generovanie farebných notových zápisov pre xylofón
Generation of music notation for xylophone

Anotácia: Pre výučbu hry na xylofón sa používa zápis formou farebných nôt, rozne xylofóny môžu mať rôznu farebnosť a farby v notovom zápise nemusia sedieť. Cieľom práce bude previesť notový zápis z midi formátu na zápis s farbami zodpovedajúcimi konkrétnemu xylofónu. Aplikácia bude tiež umožňovať automatickú extrakciu farieb jednotlivých tónov z fotografie xylofónu.

Vedúci: RNDr. Zuzana Berger Haladová, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 13.10.2022

Dátum schválenia: 24.10.2022 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné prehlásenie: Čestne prehlasujem, že som túto prácu vypracovala samostatne iba s použitím uvedených zdrojov a na základe konzultácií so školiteľkou.

. . . . . .

Poďakovanie: Ďakujem mojej školiteľke RNDr. Zuzane Berger Haladovej, PhD., za jej rady, odbornú pomoc, nápady pri práci, trpezlivosť a ochotu. Taktiež ďakujem mojej rodine a priateľom za ich nekonečnú podporu.

Abstrakt

Cieľom tejto bakalárskej práce je vytvoriť pythonovskú aplikáciu, ktorá dokáže spracovať fotografiu xylofónu, extrahovať farby jednotlivých farebných drievok, spracovať notový zápis midi formátu a vďaka získaným dátam vytvoriť vlastný farebný notový zápis. V práci je pre čitateľa vysvetlený úvod do hudobnej teórie a opísaná ďalšia potrebná teória k pochopeniu použitých technológií. Je v nej popísaný návrh postupu pri spracovaní obrazu, notového zápisu v midi formáte a nasledovného vývoja aplikácie.

Kľúčové slová: spracovanie obrazu, xylofón, midi formát, farebný notový zápis

Abstract

The aim of this bachelor thesis is to develop a Python application capable of processing a photograph of a xylophone, extracting the colors of the individual colored bars, processing musical notation in MIDI format, and using the acquired data to create a customized colored musical notation. The thesis provides an introduction to music theory and explains the additional theory necessary for readers to comprehend the utilized technologies. It outlines the design approach for image processing, handling musical notation in MIDI format, and the subsequent development of the application.

Keywords: image processing, xylophone, MIDI format, colored musical notation

Obsah

Zoznam obrázkov

Úvod

Xylofón je hudobný nástroj skladajúci sa zo sústavy ladených drievok, na ktoré sa hrá udieraním paličky vyrobenej z dreva či plastu. K výučbe hry na xylofóne sa používa notový zápis pomocou farebných nôt, vďaka ktorému vedia deti - ktorým sú tieto noty v prvom rade určené - hre a nástroju lepšie a rýchlejšie porozumieť.

Daný notový zápis sa výlučne vzťahuje na xylofón, ktorého drievka farebne zodpovedajú práve tomuto notovému zápisu. Na trhu s hudobnými nástrojmi sa ale nachádza veľké množstvo rozdielnych xylofónov s rozličnými tónovými rozsahmi i farebnosťou drievok. V prípade, že užívateľ xylofónu vlastní xylofón, ktorý farebne nezodpovedá tomuto notovému zápisu, samotný princíp ľahko pochopiteľného notového zápisu stráca na význame.

Aby boli farebné noty použiteľné aj pre užívateľov, ktorí nevlastnia konkrétne daný model, ktorý tomuto notovému zápisu farebne zodpovedá, je potrebné upraviť farebnú štruktúru notového zápisu tak, že bude farebne zodpovedať užívateľovmu xylofónu. Pracným a časovo náročnejším prepisom a prekreslením notového zápisu by sa síce zachovala ľahkosť pochopenia nôt, ale znížila by sa ekeftivita práce a hry. Zdigitalizovaním notového zápisu, naskenovaním xylofónu a pomocou automatického spracovania by zostala zachovaná jednoduchosť a efektivita čítania nôt a užívania tohto hudobného nástroja.

Cieľom tejto práce je teda vytvoriť práve takýto program, ktorý v prvom kroku spracuje fotografiu xylofónu, z ktorej získa farby a priradí ich jednotlivým tónom, a následne v druhom kroku spracuje notový zápis z midi formátu na taký notový zápis, ktorý bude farebne zodpovedať už spomenutému xylofónu. Prevod z tlačenej formy notového zápisu do midi formátu je mimo záber tejto práce a venuje sa mu iná bakalárska práca.

Práca je rozdelená na štyri kapitoly. V prvej kapitole je vysvetlená teoretická časť práce, ktorá je potrebná pri tvorbe programu.

Druhá časť je venovaná návrhu jednotlivých krokov pri práci a tretia už samotnej tvorbe aplikácie. Sú tu spomenuté použité technológie i opísaný postup.

V poslednej kapitole je opísaná funkčnosť aplikácie a vysvetlené jej ovládanie.

Kapitola 1

Analýza problému

V tejto kapitole sa zoznámime s potrebnou hudobnou i technickou teóriou a technológiami využitými v mojej práci.

1.1 Potrebná teória

V tejto časti kapitoly sú prípadnému čitateľovi opísané potrebné zákadné znalosti, ktoré potrebuje mať pri čítaní tejto práce.

1.1.1 Základná hudobná teória

Táto časť je venovaná iba najzákladnejšej hudobnej teórii, ktorá je potrebná k pochopeniu farebného notového zápisu, s ktorým budeme neskôr v tejto práci pracovať.

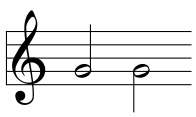
Na interpretáciu a zápis hudobnej skladby je potrebné notové písmo. Je to súbor grafických znakov, skratiek, značiek a slovných výrazov. Grafické znaky, ktorými vieme zapísať do notovej osnovy výšku a dĺžku tónu nazývame noty. [6]

Notová osnova

Na zapisovanie nôt používame notovú osnovu, ktorá sa skladá z piatich čiar a štyroch rovnako veľkých medzier.

Tvary nôt

Notové znaky sa skladajú z hlavičky, nožičky a zástavky. Nožičky sa zapisujú kolmo na notovú osnovu, a to buď smerom nahor po pravej strane hlavičky, alebo po ľavej strane smerom nadol. V prípad, že nota obsahuje aj zástavku, táto sa vždy píše vpravo.

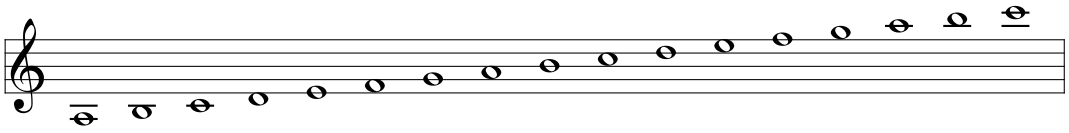


V súčasnosti sa používajú tieto tvary nôt:

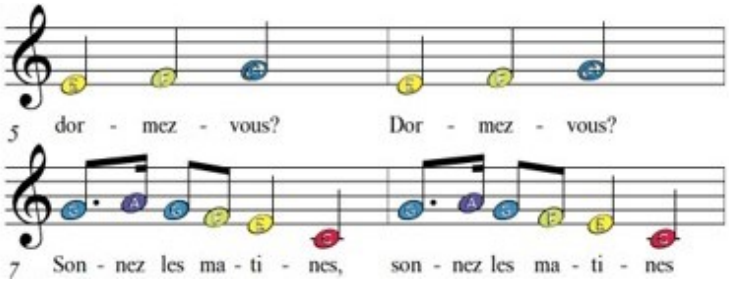
- 𝅝 - celá
- 𝅗𝅥 - pólová
- ♩ - štvrťová
- ♪ - osminová
- 𝅘𝅥𝅯 - šestnástinová
- 𝅘𝅥𝅰 - dvaatridsatinová

Keďže na zápis všetkých nôt nestačí rozsah notovej osnovy, preto používame krátke pomocné čiary pod a nad notovou osnovou pre každú notu zvlášť, pričom medzery medzi čiarami sú rovnako veľké ako medzery v notovej osnove. Používame maximálne tri až štyri pomocné čiary. Pri vyššom počte pomocných čiar by sa čítanie nôt stalo náročnejším a menej prehľadným. Tieto pomocné čiary počítame od notovej osnovy, čiže najbližšie k notovej osnove hore a dole je prvá pomocná čiara.

Na začiatku notovej osnovy je zapísaný kľúč (značka), ktorý udáva umiestnenie jednej noty, podľa ktorej vieme vypočítať výšku ostatných nôt. Husľový kľúč označuje, že na druhej čiare od spodu leží G. Zvyšné noty sa podľa tejto noty počítajú a píšu na čiary a do medzier podľa notovej abecedy.

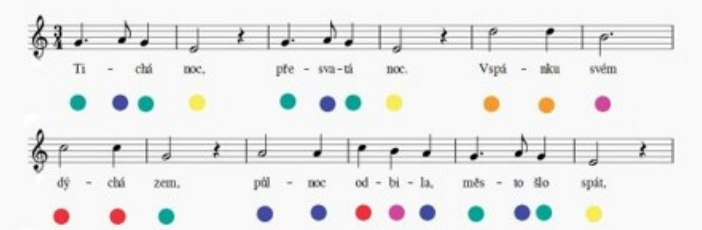


Farebné notové zápisy, s ktorými sa v tejto práci pracuje a ktoré je potrebné upraviť podľa zodpovedajúceho xylofónu, majú rôzne tvary.

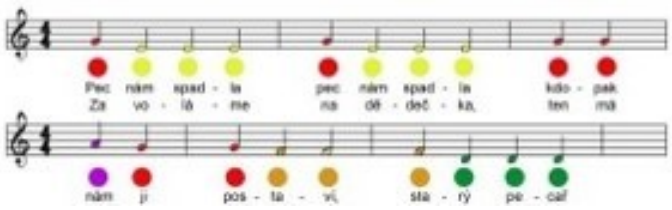


Obr. 1.1: Ukážka farebného notového zápisu

Niektorých notový zápis sa skladá z jednoduchého vloženia farebných guličiek do notovej osnovy (rovnako ako by do nej boli zapísané klasické noty); iné majú tvar štandardného notového zápisu, pričom jednotlivé noty sú zafarbené podľa tónu, ktorý má hráč zahrať (obrázok 1.1); potom sú také, ktoré majú pod notami, resp. notovou osnovou, farebnú gulličku (obrázok 1.2); prípadne sú i také, ktoré sú rôznymi kombináciami predchádzajúcich možností (obrázok 1.3).

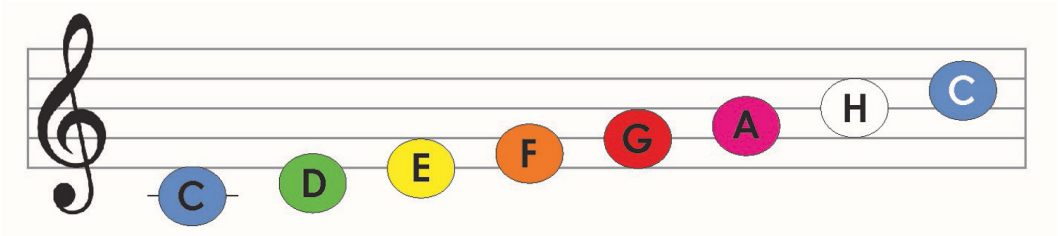


Obr. 1.2: Ukážka notového zápisu s farebnými guličkami pod notovou osnovou



Obr. 1.3: Ukážka farebného notového zápisu s farebnými guličkami pod notovou osnovou

V tejto práci pracujeme s prvou formou týchto farebných nôt, ktorú môžme vidieť na obrázku 1.4.



Obr. 1.4: Ukážka farebného notového zápisu v podobe farebných guličiek namiesto nôt

1.2 Prehľad technológii

V práci je aktívne využívaná technológia OpenCV a práca s MIDI, preto je potrebné sa s týmito pojmami zoznámiť, na čo slúži práve táto časť kapitoly.

1.2.1 MIDI a SMF

Význam slova MIDI, ktoré je skratkou pre Musical Instrument Digital Interface, môže byť rôzny. Môže sa vzťahovať na hardvérové rozhranie, formát súboru, dáta v štandardnom MIDI súbore, alebo na špecifikáciu simulácie hudobných nástrojov v rámci všeobecného MIDI. [10]

MIDI je štandardný komunikačný protokol, ktorý sa používa na ovládanie elektronikých hudobných nástrojov. Umožňuje uloženie dát (podrobné údaje týkajúce sa výkonu a súvisiace informácie o ovládaní) a ich prenos medzi elektronickými hudobnými nástrojmi, počítačmi a ďalšími zariadeniami.

MIDI nie je zvuk, ale len informácia o tom, ako sa má hudba či zvuk vykonať.

Zatiaľ čo MIDI je štandard pre komunikáciu hudobných informácii, SMF je súborový formát pre ukladanie týchto informácii. Formát súboru SMF je jedným z populárnych formátov digitálnej hudby.[12]

Nižšie bližšie opíšem MIDI a SMF, ktoré sú pre túto prácu podstatné.

MIDI

MIDI skladá z hardvérového rozhrania a komunikačného protokolu. Používa sa na ovládanie elektronických hudobných nástrojov; pomocou hardvérového rozhrania spája cez prenosové linky elektronické hudobné nástroje, počítače a iné digitálne zariadenia. Jeho príkazy alebo správy o hraní hudby sa tak môžu prenášať.

MIDI správa sa skladá z jedného bajtu stavu a niekoľkých bajtov dát. Bajt stavu udáva typ prenášanej správy, nasledovaný dátovými bajtmi súvisiacimi s touto správou. Rôzne správy MIDI môžu mať rôzne počty dátových bajtov. [12]

SMF

Standard MIDI File, skrátene SMF, je súborový formát určený na uloženie hudobných informácii v MIDI formáte.

Na rozdiel od iných formátov, v ktorých je zvuk uložený digitalizovaný, tento súborový formát umožňuje ukladanie do súboru MIDI dáta, ako sú napríklad informácie o použitých nástrojoch, tónoch, tempe, dynamike a čase hudby.

SMF umožňuje uchované dáta jednoducho prenášať medzi zariadeniami a platformami a neskôr ich použiť na prehrávanie v rôznych hudobných softvéroch, ktoré tento formát podporujú, alebo úpravu v nejakom hudobnom editore.

Vďaka malej veľkosti súboru majú SMF široké využitie; sú využívané v počítačoch, tónoch vyzvánania mobilných telefónov či vo webových stránkach.

SMF sa skladá z viacerých častí využívaných na prehľadné ukladanie hudobných informácií. Hlavička a stopa sú dva typy záznamov v štruktúre SMF. Štruktúra SMF

vždy začína hlavičkou (v každom súbore je len jedna hlavička) a nasleduje niekoľko stôp. Tieto stopy tvoria hudobný súbor.

Hlavička aj stopa začínajú štyrmi ASCII znakmi, ktoré označujú o aký typ záznamu ide: MThd pre hlavičku (skratka pre Main Header), MTrk pre stopu (MIDI Track), a pokračujú informáciami o dĺžke 32 bitoch, ktoré sa používajú na znázornenie, koľko bytov údajov je vo zvyšku daného záznamu.

Header chunk				
MThd	Length of Header Data	Format	Number of Tracks	Division

Hlavička obsahuje informácie o čísle stopy, verzii formátu, delení, dĺžke a o ďalších dôležitých informáciach o SMF. Dĺžka delenia zaberá 16 bitov a poznáme dva typy formátov:

- ak je hodnota MSB (Most Significant Byte) delenia 1, určuje to, že delenie je formátu time-code-based (založenom na časovom kóde), často používanom pri synchornizácii medzi video a MIDI zariadeniami

- ak je táto hodnota 0, znamená to, že toto delenie určuje počet tikov štvrťovej noty, pričom tik je najmenšia jednotka času v SMF súboroch. Ide o formát, ktorý je vo všeobecných SMF bežne používaným formátom delenia.

Hodnota delenia priamo súvisí s dĺžkou aktuálneho času, ktorý zodpovedá hodnotám delta času.

Track chunk						
MTrk	Length of Tracker Data	Delta Time	Evend 1	...	Delta Time	Event n

Zrejme najpodstatnejšou časťou je MTrk. Záznam stopy sa používa na uloženie skutočných informácií o hudbe, ako napríklad o aký hudobný nástroj sa jedná, tóny, ich rýchlosť, dynamika a časové údaje.

Delta čas je časový interval medzi dvoma udalosťami, pričom ak je hodnota delta času 0, znamená to, že dve udalosti sa vyskytujú naraz.

Časový údaj sa skladá z dvoch čísel zapísaných v hudobnom zápise. Usporiadané sú podobne ako zlomok v matematike - čitateľ označuje počet úderov za takt, zatiaľ čo menovateľ určuje hodnotu tónu úderu, čiže tón ktorý sa pri údere používa. Časový údaj SMF je nastavený udalosťou časového údaja.

1.2.2 OpenCV

OpenCV je open source knižnica počítačového videnia, napísaná v programovacích jazykoch C a C++. Táto knižnica poskytuje na použitie jednoduchú infraštruktúru počítačového videnia, vďaka ktorej je možné rýchlo vytvárať aj náročné aplikácie. Funkcie, ktoré táto knižnica poskytuje, pokrývajú množstvo oblastí videnia, ako napríklad pri kontrole produktov vo výrobe, bezpečnosti, lekárskom zobrazovaní, stereo videní, kalibrácie kamery či robotiky. [5]

OpenCV podporuje množstvo programovacích jazykov, ako napríklad Python, C++, Java, atď., je k dispozícii na rôznych platformách vrátane Windows, Linux, OS X, Android a iOS [1].

V tejto práci je používaná OpenCV verzia určená pre Python.

1.2.3 Kvantizácia

Kvantizácia farieb obrazu je proces, ktorý redukuje počet farieb [9]. Tento proces zahŕňa výber špecifickej množiny farieb, ktoré predstavujú a reprezentujú farebnú škálu na obrázku. Zahŕňa aj výpočet mapovania, ktorý spája farebný priestor s týmito vybranými reprezentatívnymi farbami. [7].

Pri kvantizácii obrazu sa teda každému pixelu v obraze priradí jedna z diskrétnych hodnôt, ktoré reprezentujú rozsah farebných alebo jasových hodnôt.

Vo väčšine existujúcich metód kvantizácie farieb sa vyžaduje, aby používateľ manuálne vybral počet farieb na konečnom obrázku. Metóda popísaná v [9] však automaticky vypočíta optimálny počet farieb v obraze s cieľom čo najviac zachovať kvalitatívny popis obrazu [9]. Nadmerná kvantizácia môže spôsobiť viditeľné artefakty a stratu informácii.

Metód, ktoré umožňujú výber týchto reprezentatívnych farieb je viacero, v tejto práci bola využitá metóda K-Means [4].

1.2.4 Perspektívna transformácia

Perspektívna transformácia umožňuje projekciu jednej roviny do druhej. Slúži na simuláciu pohľadu z rôznych perspektív či uhlov. Využíva sa napríklad na zobrazovanie 3D scény v 2D priestore.

Perspektívna transformácia je založená na Homografickej matici, ktorá má rozmer 3x3. Na výpočet hodnôt Homografickej matice je potrebné mať minimálne štyri páry zodpovedajúcich bodov vstupného a výstupného obrazu [8]. Na základe týchto bodov sa vypočíta a aplikuje transformačná matica.

V tejto práci dané vybrané body zodpovedajú hranám xylofónu, resp. vonkajším hranám krajného pravého a ľavého drievka xylofónu.

1.2.5 Houghova transformácia

Houghova transformácia je algoritmus využívaný v oblasti počítačového videnia a spracovania obrazu na detekciu priamok.

Pre Houghovu transformáciu budeme vyjadrovať priamky v polárnom súradnicovom systéme pomocou parametrov (r, θ) [11]. Rovnicu každej priamky vieme zapísať nasledovne:

$$y = (-\frac{\cos\theta}{\sin\theta})x + (\frac{r}{\sin\theta}) \tag{1.1}$$

Úpravou dostaneme:

$$r = x \cdot \cos\theta + y \cdot \sin\theta \tag{1.2}$$

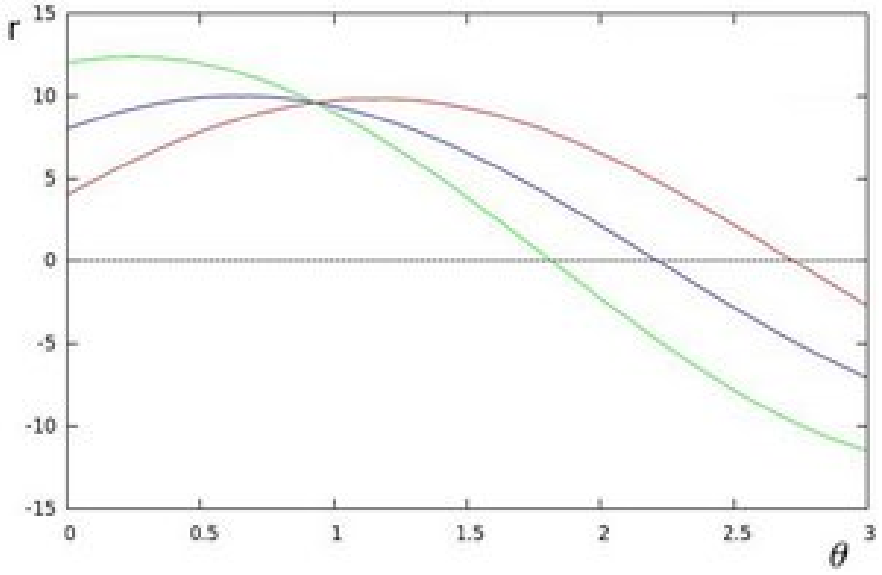
Všeobecne platí, že pre každý bod (x_0, y_0) môžeme identifikovať množinu priamok, ktoré ním prechádzajú. Tieto priamky môžeme vyjadriť pomocou vzťahu:

$$r_\theta = x_0 \cdot \cos\theta + y_0 \cdot \sin\theta \tag{1.3}$$

pričom jeden pár (r_θ, θ) reprezentuje jednu priamku, ktorá prechádza bodom (x_0, y_0).

Ak by sme vykreslili množinu všetkých priamok, ktoré prechádzajú bodom (x_0, y_0), dostaneme sínusoidu. Takto postupujeme každým bodom obrazu.

To, že sa dva body nachádzajú na jednej priamke vieme podľa toho, že sa sínusoidy dvoch rôznych bodov pretínajú v rovine θ-r.



Obr. 1.5: Pretínajúce sa sínusoidy [2]

Čím viac sínusoid sa pretína v tom istom bode, tým viac bodov, ktoré ležia na tejto priamke. Priamku môžeme detekovať na základe prekročenia definovaného minimálneho počtu pretínajúcich sa bodov.

Táto transformácia sa vykonáva na binárnom obraze, ktorý sa získava spracovaním pôvodného obrazu aplikáciou detektora hrán.

1.3 Podobné práce

V súčasnosti neexistuje žiadny komplexný projekt alebo program, ktorý by riešil problematiku tejto práce. Voľne dohľadateľné sú len čiastkové riešenia, programy alebo štúdie, ktorých súčasťou sú aj riešenia zameriavajúce sa na jednotlivé časti, ktoré sú v tejto práci riešené.

Napríklad existujú programy, ktoré sa zameriavajú na rozoznanie jednotlivých hudobných nástrojov z obrazu, alebo práce, ktoré sa snažia o digitalizovanie nôt.

Všetky tieto práce sa ale zameriavajú na vyriešenie omnoho komplexnejšieho problému - či už rozpoznávanie viacerých hudobných nástrojov alebo nôt vo všeobecnosti (nie len ich jednoduchšiu verziu vo farebnej forme, ako to je v tejto práci).

Kapitola 2

Návrh

Cieľom tejto práce je vykonávať konverziu notového zápisu z MIDI formátu na farebný zápis zodpovedajúcimi konkrétnemu xylofónu, pričom farby jednotlivých tónov budú automaticky extrahované z fotografie xylofónu.

Proces tejto konverzie vieme rozdeliť na dve časti:

1. Extrakcia farieb jednotlivých tónov z fotografie xylofónu
2. Prevod notového zápisu z MIDI formátu na farebný zápis (v tejto časti sa využívajú extrahované farby tónov z predchcádzajúcej časti

Obe tieto časti v tejto kapitole podrobne popíšeme.

Vstupnými údajmi, ktoré slúžia ako zdroj informácií pre konverziu pre python aplikáciu sú:

- fotografia xylofónu
- súbor vo formáte SMF (Standard MIDI File) alebo textový súbor

Výstupom aplikácie sú obrázky s farebným notovým zápisom, ktoré zobrazujú prevedené noty v súlade s farebným kódom xylofónu a textový súbor obsahujúci jednoduchý notový zápis.

Pre realizáciu tejto práce budú využité nasledujúce knižnice: *OpenCV* na spracovanie obrazu, *music21* na prácu s notovým zápisom a *PyQt* pre vytvorenie užívateľského rozhrania.

2.1 Získavanie farieb tónov

Na vygenerovanie nôt, ktoré budú farebne zodpovedať nejakému konkrétnemu xylofónu budeme potrebovať fotografiu tohto xylofónu, pričom predpokladáme, že pozadie na forografii bude jednofarebné. V tejto časti je opísaný postup spracovania fotografie

xylofónu s cieľom získať nejakú reprezentatívnu farbu pre každý tón, resp. každé toto drievko, a následná extrakcia týchto farieb.

2.1.1 Perspektívna transformácia

Po načítaní fotografie xylofónu na nej vyberieme štyri body, vďaka ktorým označíme rohy xylofónu, resp. vonkajšie hrany prvého (ľavého) a posledného (pravého) drievka xylofónu. Pomocou týchto štyroch bodov sa vypočíta transformačná matica a aplikuje perspektívna transformácia.

Vďaka perspektívnej transformácii získame orezaný obrázok, vďaka čomu minimalizujeme rôzne skreslenia a možné problémy v nasledujúcich krokoch spracovania forografie xylofónu.

2.1.2 Kvantizácia

V tejto časti práce sme pomocou kvantizácie znížili počet farieb na orezanej fotogragrafii so xylofónom. Počet zredukovaných farieb predstavoval počet drievok xylofónu +1 (farba určená pre pozadie za, resp. pod xylofónom). Pre výber týchto reprezentatívnych farieb sme použili metódu K-Means.

Pôvodný obrázok sme najprv prekonvertovali do formátu RGB a naň, s cieľom získať daný počet reprezentatívnych farieb, aplikovali algoritmus K-Means, ktorý umožňuje priradiť každý pixel k jednej z daných farieb na základe ich podobnosti.

Po získaní farebnej palety sme nahradili farby v pôvodnom obrázku farebnými hodnotami z tejto palety farieb. Takto sme dosiahli redukciu počtu farieb na požadovaný počet a vytvorili tak nový obrázok, kde sú všetky farby nahradené jednou z farieb získanej palety.

2.1.3 Detekcia hrán a priamok

Kvantizovaný obrázok prevedieme na šedotónový. Následne aplikujeme mediánový filter na redukovanie prípadného šumu v obrázku. V tejto chvíli je cieľom nájsť hrany jednotlivých drievok xylofónu.

Prahovanie

Na záskanie hrán drievok xylofónu použijeme adaptívnu prahovú metódu, vďaka čomu získame binárny obraz, v ktorom sú tieto hrany (a niektoré ďalšie časti xylofónu, ako napríklad okraje *klinčekov*) biele a pozadie a vnútro drievok čierne. Ďalším krokom je detekcia hrán z tohto pomocou Cannyho algoritmu, ktorý umožňuje identifikáciu hrán objektov v obraze.

Houghova transformácia

Nakoniec je s cieľom nájdenia priamok v obraze aplikovaná Houghova transformácia. Po tejto operácii získame množinu priamok,z ktorých niektoré ležia na hranách drievok xylofónu. Keďže sú drievka na xylofóne uložené vertikálne vedľa seba,z týchto nájdených priamok nás zaujímajú iba tie vertikálne.

Na základe hodnoty θ vypočítame uhol, ktorý predstavuje sklon danej priamky. Na to, aby bola priamka považovaná za vertikálnu, musí byť jej sklon blízky 0° alebo 180° (smerom pozdĺž osi y). Do zoznamu vertikálnych priamok pridáme iba tie, ktorých sklon spĺňa túto podmienku s toleranciou 10°.

2.1.4 Lokalizácia drievok xylofónu

Po získaní množiny vertikálnych priamok potrebujeme označiť a pracovať iba s tými priamkami, ktoré označujú hrany drievok xylofónov - konkrétne páry priamok, ktoré tvoria pravé a ľavé hrany daných drievok. Na toto sme priamky najprv zoradili tak, aby sme ich čítali po sebe a postupne počítali vzdialenosť medzi nimi.

Pri pohľade na forografiu xylofónu upravenú pomocou perspektívnej transformácie môžme vidieť, že obrázok tvoria z najväčšej časti drievka xylofónu a medzery medzi nimi sú menšie ako šírka drievok. Na základe tohto vieme predpokladať, že ak pre xylofón, ktorý sa skladá z 8 drievok, nájdeme osem párov priamok medzi ktorými sú najväčšie medzery, pričom priamky z tejto dvojice sa na obraze nachádzajú vedľa seba, získame práve tých osem párov priamok, ktoré označujú hrany drievok xylofónu. Týmto prístupom sa vyhneme aj problému, kde na jednej hrane drievka sme doteraz našli viac ako len jednu priamku, keďže z týchto priamok sa bude počítať vždy len jedna.

2.1.5 Extrakcia farieb

Na extrakciu farieb tónov sme postupovali nasledovne: Pre každé drievko sme identifikovali dve krajné priamky, pričom sme určili stred medzi nimi na osi x. Následne sme vypočítali stred obrázka na osi y. Na základe týchto hodnôt sme lokalizovali konkrétny pixel a získali tak jeho farbu. Keďže bola na obrázku vykonaná kvantizácia, počet farebných odtieňov bol zredukovaný. Týmto prístupom sme dosiahli väčšiu istotu, že každý pixel bude presne zodpovedať správnej farbe,a teda farbe zodpovedajúcej danému tónu, resp. drievku.

2.2 Prevod notového zápisu z MIDI

Druhým cieľom tejto práce je previesť notový zápis z midi formátu na zápis s farbami, ktoré zodpovedajú danému xylofónu.

Na konci nižšie spomínaného procesu sú na výstupe PNG súbory s farebným notovým zápisom a textový súbor s jednoduchým notovým zápisom.

2.2.1 Spracovanie súboru v MIDI formáte

Pri spracovaní súboru v midi formáte (SMF), sme využívali knižnicu music21, ktorá poskytuje funkcie, pomocou ktorých môžme pracovať s hudobnými dátami. Na prehrávanie a vytváranie notových zápisov sme využili nástroj MuseScore.

Po načítaní súboru do programu sme ho postupne čítali, pomocou knižnice music21 vytvorili zoznam nôt a tieto informácie o notách zapisovali do textového súboru, ktorý bude čítaný v nasledujúcom kroku.

Pre tento postup sme sa rozhodli, pretože okrem čítania notového zápisu z midi formátu sme chceli pre úžívateľa pridať možnosť vytvoriť si vlastný notový zápis - a to čo najjednoduchšou cestou.

2.2.2 Generovanie nôt

Po vytvorení textového súboru a zápisu nôt, sa tento súbor prečíta a vygenerujú sa farebné noty, ktoré zodpovedajú xylofónu, a ktoré sme získali v prvom kroku tejto práce.

Po naštudovaní si knižníc, ktoré poskytujú možnosť generovania farebných nôt, sme vyhodnotili, že bude najlepšie si vytvoriť vlastnú triedu na generovanie farebných nôt, ktoré budú spĺňať naše očakávania. K tomuto kroku sme dospeli najmä preto, že sme chceli získať čo najčitateľnejší notový zápis, s ktorým by deti nemali problém, a ktorý by sa čo najviac podobal vyššie uvedeným príkladom.

Ako výstup sme vytvorili PNG súbor, v ktorom sa zobrazia noty načítané zo SMF, alebo textového súboru, pričom tieto noty sú zobrazené so správne priradenými farbami.

Kapitola 3

Implementácia

V tejto kapitole by sme sa chceli zamerať na implementáciu projektu a podrobne opísať kroky, ktorými sme postupovali pri vývoji aplikácie. Hlavným súborom nášho projektu je *main.py*. Na dosiahnutie vyššie stanovených cieľov využívame rôzne knižnice, ako *OpenCV, NumPy, PyQT, OS, shutil, PIL*.

3.1 Lokalizácia drievok xylofónu a extrakcia farieb

V tejto časti sa budem venovať spracovaniu fotografie so xylofónom s cieľom získania farieb a ich priradeniu k jednotlivým tónom. Na toto sme si vytvorili vlastnú triedu *FindColors()* (súbor *ColorsFinder2.py*).

3.1.1 Perspektívna transformácia

Predpokladáme, že vstupný obrázok xylofónu bude dostačne ostrý a pozadie bude jednofarebné. Po výbere a načítaní fotografie xylofónu na nej vyberieme štyri body, ktorými označíme rohy xylofónu v takomto poradí: ľavý horný roh, pravý horný roh, ľavý dolný roh a nakoniec pravý dolný.

Pomocou týchto štyroch bodov sa za použitia funkcie knižnice *OpenCV getPerspectiveTransform* vypočíta transformačná matica. Perspektívna transformácia sa vykoná pomocou *warpPerspective*, ktorá je taktiež funkciou knižnice *OpenCV*.

Na obrázku vidíme, že po transformácii došlo k deformácii xylofónu, čo ale neovplyvňuje výsledky v nasledujúcich krokoch tejto práce, keďže, ako sme už vyššie spomenuli, pri hľadaní hrán jednotlivých drievok nás budú zaujímať ich zvislé hrany.

Vďaka tejto úprave obrázka sme minimalizovali monžé problémy v nasledujúcich krokoch - ako napríklad prípady, kedy by sme našli *falošné* hrany v ráme xylofónu; či narazili na iné nepredvídateľné situácie (napríklad ak by pozadie fotografie bolo viacfarebné či vzorované, v takomto prípade sme pravdepodobnosť problému s týmto pozadím aspoň minimalizovali).

Obr. 3.1: Fotografia xylofónu s označenými rohmi

Obr. 3.2: Perspektívna transformácia použitá na fotografii xylofónu

3.1.2 Kvantizácia

Implementácia algoritmu K-Means [4] v našom systéme prebieha nasledovne:

Ako sme už vyššie spomínali, obrázok sme previedli z pôvodného farebného formátu BGR na RGB, a to pomocou funkcie *cvtColor()* z knižnice *OpenCV*. Podľa [4] má byť obrázok prevedený na formát *float32*. Následne ho normalizujeme v rozsahu od 0 po 1. Zložky farieb jednotlivých pixelov obrazu sú pretransformované na jednorozmerné pole pomocou metódy *reshape()*. Definujeme kritéria pre ukončenie algoritmu, ktoré nastavíme na dosiahnutie maximálneho počtu iterácii (10) alebo zmeny centroidov menšej ako nastavená hodnota epsilon (1.0). Pre charakter vstupného obrázku a výsledky, ktoré sme pri testovaní získali, sme usúdili, že nie je potrebný vyšší maximálny počet iterácii. Týmto výberom sme dosiahli optimálnu kombináciu pre náš problém.

Premennú K sme nastavili na hodnotu *number_of_notes + 1*, a teda na počet drievok xylofónu + 1 . Táto hodnota určí počet zhlukov (clusters), do ktorých budú vstupné dáta rozdelené, pričom počet týchto zhlukov zodpovedá počtu hľadaných tónov, resp. farieb drievok + 1 vyhradené pre farbu pozadia.

Algoritmus K-Means sme aplikovali pomocou metódy *kmeans(data, K, bestLabels, criteria, attempts, flags[, centers])* z knižnice OpenCV aplikovaný na jednorozmerné pole s počtom zhlukov *K*. Parameter *bestLabels* (voliteľný a slúžiaci na poskytnutie počiatočných priradení zhlukov) sme nastavili na *None*, čo znamená, že pre počiatočné centroidy sa použije náhodné rozdelenie.

Parametru *attempts* sme zvolili hodnotu 10, ktorá predstavuje počet pokusov, ktoré algoritmus k-means vykonáva s rôznymi počiatočnými centroidami, pričom pri každom z týchto pokusov sa algoritmus spustí s inými náhodne vybranými počiatočnými centroidami a vypočíta príslušné priradenia zhlukov a stredové hodnoty. Vyšší počet pokusov môže síce zvýšiť pravdepodobnosť priaznivejších výsledkov, zároveň ale zvyšuje výpočtovú náročnosť algoritmu.

Premennej *flags* sme zvolili KMEANS_RANDOM_CENTERS, ktorá definuje spôsob inicializácie počiatočných centroidov pre algoritmus k-means clustering. Pri použití tejto hodnoty sa počiatočné centroidy zhlukov vyberajú zo vstupných dát náhodne.

Výstupom získame pole *labels*, ktoré obsahuje indexy zhlukov pre každý prvok v jednorozmernom poli, a pole *centers*, ktoré obsahuje stredové hodnoty (centroidy) pre každý zhluk.

Následne sú farebné hodnoty v obrázku nahradené ich priemernými hodnotami (centroidami) zo zhlukov a obrázok je konvertovaný späť na farebný formát BGR, rovnako pomocou funkcie *cvtColor()* z knižnice OpenCV.

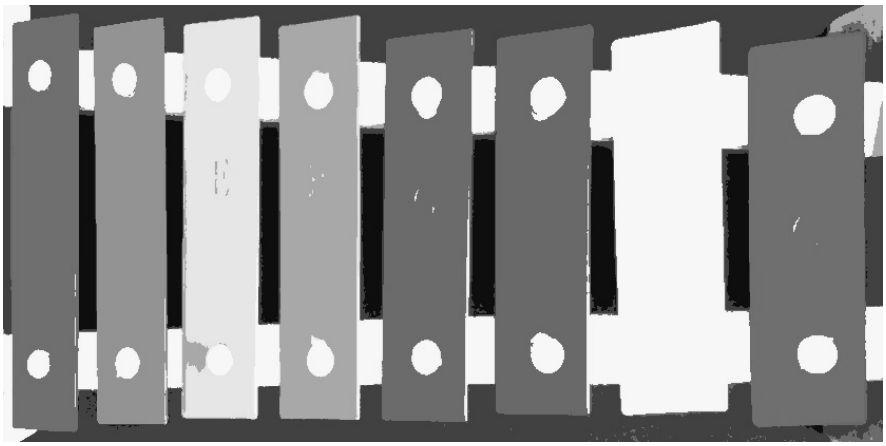
Obr. 3.3: Fotografia xylofónu po perspektívnej transformácii a kvantizácii

3.1.3 Odstránenie šumu

V tejto časti by sme chceli zhrnúť ďalšie spracovanie obrazu s cieľom pripraviť obrázok pre detekciu hran.

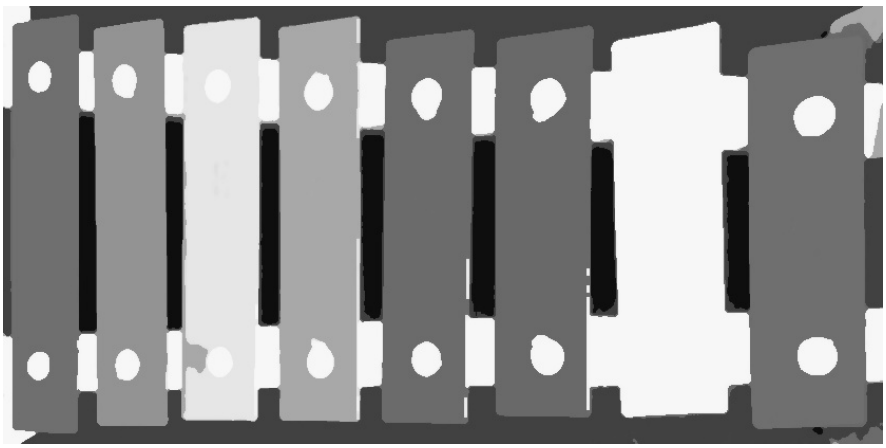
Na začiatku spracovania je potrebné načítať vstupný obrázok - fotografiu xylofónu po perspektívnej transformácii a kvantizácii. V prvom kroku sme si previedli obraz do

odtieňov šedej. Na tento účel používame funkciu *cvtColor(image, cv.COLOR_BGR2GRAY)*, ktorá transformuje obrázok na čiernobielu verziu.



Obr. 3.4: Šedoúrovňové ozbrazenei obrázka

Na redukciu potenciálneho šumu v obraze a na získanie hladšieho výsledku, aplikujeme na obrázok mediánový filter pomocou funkcie *medianBlur(src,ksize[, dst])*. Ako druhý parameter *ksize*, ktorý určuje veľkosť kernelu, v ktorom sa vykoná vyhladenie, je číslo 5, čo sa ukázalo ako vhodná hodnota pre úlohu spracovania takéhoto obrázka.



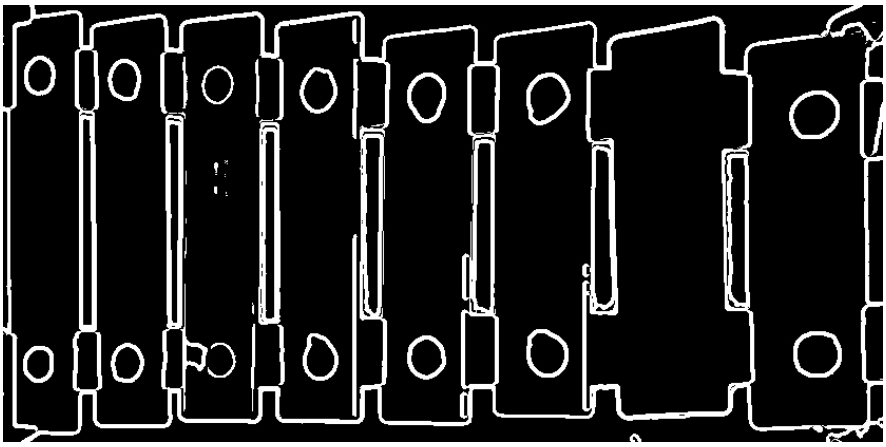
Obr. 3.5: Obrázok po aplikácii mediánového filtra

3.1.4 Prahovanie

V ďalšom kroku aplikujeme na vyhladený obrázok adaptívne prahovanie, na čo použijeme metódu *adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst])* [3]. *MaxValue*, alebo maximálnu hodnotu prahu, ktorá sa priradí tým pixelom, ktoré prekročia prahovú hodnotu, sme nastavili na 255. Ako metódu adaptívneho prahovania (parameter *adaptiveMethod*) sme zvolili ADAPTIVE_THRESH_GAUSSIAN_C. Z dvojice typov prahovania (parameter *thresholdType*) sme si zvolili THRESH_BINARY_INV,

a teda inverzné binárne prahovanie, kde pixely nad prahom budú mať hodnotu 0 (čiže budú čiernej farby) a pixely pod prahom budú mať hodnotu 255 (budú bielej farby). Veľkosť okolia pixelu (parametr *blockSize*), ktorá sa používa na výpočet prahovej hodnoty pre pixel, sme zvolili na 11, čiže veľkosť 11x11 pixelov. *C*, ktoré sa odčíta od priemeru alebo váženého priemeru, aby sa získal adaptívny prah, bolo zvolené na 2.

Na základe zvolených parametrov sme pri testovaní na viacerých obrázkoch získali výsledky s najvyššou pravdepodobnosťou úspechu pri ďalšej práci s nimi.



Obr. 3.6: Obrázok po adaptívnom prahovaní

3.1.5 Získavanie hrán

Na zistenie a detekciu hrán v binárnom obrázku, ktorý sme získali v predchádzajúcom kroku, používame Cannyho detektor hrán. Aplikujeme OpenCV metódu *Canny(image, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]])*. Prah *threshold1* sme po testovaní nastavili na 50 a prah *threshold2* na 150.

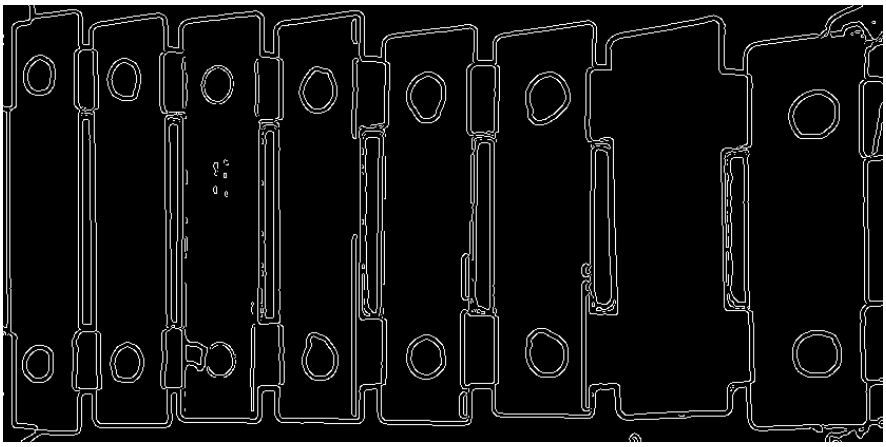
Zadané parametre sme testovali na 13 obrázkoch a vyhovovali 10 z nich. Testované fotografie xylofónu mali rozdielne farebné a svetelné podmienky a obsahovali xylofóny s rozdielnymi tónovými rozsahmi.

Výsledok detekcie hrán sme uložili do premennej *edges*.

Týmto spôsobom sme dokončili spracovanie obrázka a pripravili si ho na nasledujúci krok.

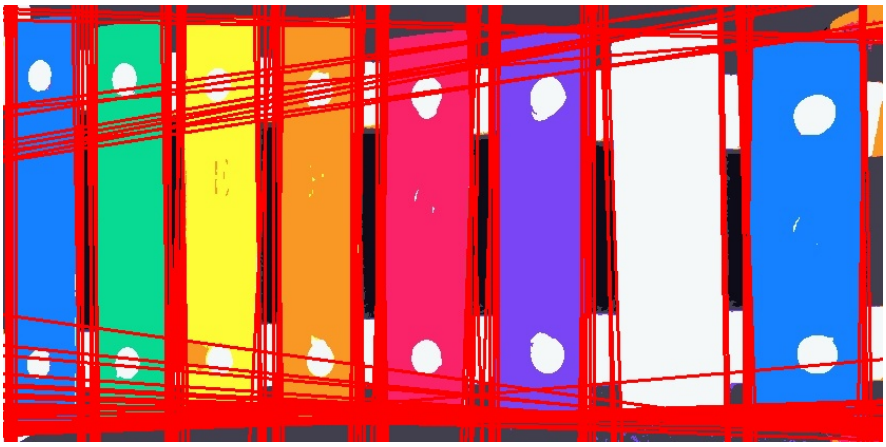
3.1.6 Houghova transformácia

Na detekciu priamok v obraze sme použili Houghovu transformáciu. Na obrázok, ktorý sme doteraz upravovali aplikujeme OpenCV metódu *HoughLines(image, rho, theta, threshold[, lines[, srn[, stn[, min_theta[, max_theta]]]]])* [2], výsledkom ktorej je zoznam detegovaných priamok v obraze.



Obr. 3.7: Zobrazenie obrázka po aplikovaní Cannyho algoritmu

Pre lepšiu prehľadnosť budeme nasledujúce výsledky zobrazovať na obrázku xylofónu po aplikácii perspektívnej transformácii a kvantizázii.



Obr. 3.8: Priamky detekované aplikovaním Houghovaj transformácie

3.1.7 Lokalizácia drievok xylofónu

Ako môžeme vidieť na obrázku, použitím Houghovej transformácie sme detekovali všetky priamky v obraze. Viacero priamok leží priamo na sebe, pretína sa, prípadne ide o priamky, ktoré vôbec nepotrebujeme.

V prvom kroku práce s nájdenými priamkami bude zlúčenie tých priamok, ktoré sú príliš blízko seba. Aby boli dve priamky považované za blízke, musia spĺňať nasledovné dve podmienky:

- maximálna povolená odchýlka vzdialenosti medzi nimi je 15 pixelov
- maximálna povolená odchýlka uhlu medzi nimi je 10 stupňov

Na vyhľadanie všetkých priamok, ktoré obe tieto podmienky spĺňajú použijeme *numpy* funkciu *where()*. Numpy funkciou *mean()* sa výpočíta priemer týchto priamok, čo vytvára novú priamku, ktorá je ich zjednotením. Táto nová priamka je uložená v premennej *merged_line*.

V druhom kroku odstránime priamky, ktoré pri pokračovaní v našej práci vôbec nepotrebujeme. Medzi takéto priamky, patria všetky také, ktoré neoznačujú zvislé hrany drievok xylofónu. Preto druhým krokom pri práci s už pretriedenými priamkami bude odstránenie všetkých priamok, ktoré nie sú vertikálne. Pri hľadaní vertikálnych priamok sme určili toleranciu na 10 stupňov.

Obr. 3.9: Zlúčené vertikálne priamky

Po odstránení všetkých prebytočných priamok z obrazu sme získali zoznam priamok, medzi ktorými budeme hľadať dvojice, ktoré označujú jednotlivé drievka xylofónu.

Ako sme vyššie spomínali, pri pohľade na obrázok xylofónu po aplikovaní perspektívnej transformácii môžeme vidieť, že jeho plochu zaberajú najmä (deformáciou xylofónu rozdielne široké) drievka xylofónu. Vďaka tomu vieme, že na lokalizáciu týchto drievok nám stačí nájsť tie páry priamok, ktoré sú od seba na osi x najviac vzdialené a neleží medzi nimi iná priamka. Napríklad, ak je drievok na xylofóne osem, hľadáme osem párov priamok, ktoré sú na osi x od seba najviac vzdialené.

Pre tento postup musíme najprv zoradiť priamky a čítať ich postupne po osi x zľava doprava. Medzi takto za sebou idúcimi priamkami vypočítame vzdialenosti a uložíme si informácie o dvojiciach priamok a ich vzdialenosti do zoznamu. Následne tieto dvojice zoradíme od tých s najväčšími medzerami po dvojice s najmenšími.

Ako môžeme vidieť na obrázku, niektoré drievka xylofónu sú označené iba jedným párom priamok, iné, napriek predchádzajúcim krokom, sú stále označené viacerými priamkami. Keďže budeme počítať vzdialenosti medzi priamkami a hľadať najväčšie z nich, problém s viacerými priamkami na jednej hrane jednoducho obídeme tým, že sa z nich bude počítať iba jedna.

Na obrázku môžeme vidieť dvojice priamok, označujúce hrany drievok xylofónu.

Obr. 3.10: Výsledné označenie nájdených hrán drievok xylofónu

Modrá priamka označuje pravú hranu, ružová ľavú. V prípade, že sa jedna z dvojice priamok na obrázku neozbrazuje, ale drievka sú označené správne, znamená to, že dve susediace drievka majú spoločnú jednu z priamok (a teda sa na obrázku iba prekrývajú).

Pri testovaní našeho programu sme narazili aj na prípady, kedy niektoré drievka neboli označené žiadnou priamkou. Výhodou nášho postupu v takomto prípade je, že viaceré drievka môžu zdieľať jednu priamku ako spoločnú hranu. Vďaka tomu, že sa pri získavaní farieb pre takéto drievka vypočítal stred medzi jedho susedmi, väčšinou sme získali správne výsledky.

3.1.8 Extrakcia farieb

Samotnú extrakciu farieb sme získali nasledovným spôsobom. Ako súradnicu Y sme určili stred obrázka na y-ovej osi. X-ovú súradnicu sme pre každé drievko vypočítali ako stred na osi x medzi dvojicou priamok, ktoré označujú hrany tohto drievka. Pomocou týchto súradníc sme lokalizovali konkrétny pixel na obrázku xylofónu po aplikovaní perspektívnej transformácie a kvantizácie a získali tak farbu každého drievka.

Na spracovanie a uloženie získaných dát sme si vytvorili vlastnú triedu *DataManager()* (súbor *DataManager.py*).

3.1.9 Vyhodnotenie

Testovanie časti programu zameranej na spracovanie obrazu prebehlo na vzorke s 15 obrázkami xylofónu. Pre väčšiu časť sme dostali požadované výsledky, a teda správne extrahované farby.

Hodnotenie sme realizovali na základe pozorovania farieb ľudským okom. Hodnotili sme teda schopnosť aplikácie extrahovať a priradiť farby pre nás ako úžateľa dostatočne uspokojivo. Aby sme priradenie konkrétnej farby považovali za správne, táto farba nesmela byť medzi ostatnými zameniteľná (pokiaľ teda nešlo o práve taký xylofón, ktorého farby sa po prechode medzi stupnicami opakovali) a musela byť priradená zodpovedajúcemu tónu v správnom poradí.

Je dôležité zdôrazniť, že naša hodnotiaca metóda sa zakladala na subjektívnom vnímaní farieb zo strany užívateľa. Hlbšie a detailnejšie vyhodnotenie tejto aplikácie by mohlo byť realizované v budúcej práci, kde by sa mohli zohľadniť aj ďalšie faktory a aspekty.

Taktiež by sme chceli spomenúť, že nami vyvinutá aplikácia je poloautomatická, keďže vie užívateľ dodatočne upravovať programom extrahované a priradené farby. Užívateľ má teda vždy finálne slovo.

V nasledujúcej tabuľke uvádzame hodnotenia pre jednotlivé fotografie:

Tabuľka 3.1: Vyhodnotenie správnosti extrahovania a priradenia farieb

Číslo xylofónu	**Počet správne priradených farieb**	**Počet očakávaných farieb**
1	8	8
2	7	8
3	7	8
4	8	8
5	8	10
6	6	8
7	8	8
8	12	12
9	5	8
10	10	12
11	8	8
12	15	15
13	12	15
14	8	8
15	7	8

Problémy

Najväčšie problémy pri spracovaní týchto fotografií spôsobili svetelné podmienky, a teda tiene či príliš presvetlené časti po použití blesku pri vytváraní fotografie. Práve pre tiene a výrazne osvetlené časti xylofónu dochádzalo pri aplikovaní kvantizácie k chybám. Týmito chybami bolo napríklad zliatie farby drievka s pozadím. V iných prípadoch boli, pre slabo osvetlený xylofón, viaceré drievka označené rovnakou farbou.

Obr. 3.11: Zle osvetlená fotografia xylofónu

Obr. 3.12: Kvantizácia aplikovaná na zle osvietenú fotografiu

Taktiež sme testovali fotografiu xylofónu, ktorý bol položený na drevenom podklade (čiže obrázok nespĺňal požiadavku jednofarebného pozadia), čo spôsobilo mierne nezrovnalosti pri hľadaní hrán drievok. Napriek tomu bolo 7 z 8 farieb priradených správne.

Obr. 3.13: Nesprávne priradenie farieb tónom

3.2 Generovanie nôt

V tejto časti sa pokúsime opísať kroky, ktorými sme postupovali pri spracovávaní notového zápisu formátu midi a následnom generovaní vlastných farebných nôt. Na toto sme si vytvorili a použili vlastnú triedu *CreateNotes()* (súbor *CreateNotesFromMIDI.py*).

3.2.1 Spracovanie nôt

Pre spracovanie súboru v midi formáte (SMF), sme využívali knižnicu *music21*. Pomocou funkcie *converter.parse()* sme uložili dáta do premennej *midi_file* a následne v cykle a pomocou volania recurse() prečítali elementy *stream-ov*. Z elementov sme do nového zoznamu uložili tie, ktoré označovali noty a akordy.

Následne sme tieto údaje cyklom prečítali a zapísali do textového súboru nasledovným spôsobom:

Pomocou *pitch.name* sme získali názov noty a pomocou *pitch.octave* sme zistili jej oktávu. Do textového súboru sme do jedného riadku zapísovali názov vždy len jednej noty (čiže napríklad *D*) a číslo oktávy. Pre našu prácu sme zjednodušili zápis tak, že ak išlo o štvrtú oktávu, k note sme nepripisovali nič. Pri piatej oktáve sme pripísali číslo 2 a pri šiestej číslo 3.

Tento spôsob sme zvolili, pretože pri zápise tónov a im priradeným farbám sme postupovali rovnako, a teda bolo jednoduchšie zachovať tento postup.

V prípade, že užívateľ vybral ako vstupný súbor textový súbor, predchádzajúci krok sa preskočí s predpokladaním, že spôsob zápisu nôt v textovom súbore je správny.

3.2.2 Vytvorenie vlastných nôt

Pomocou knižnice *PIL* (Python Imaging Library) a jej príslušných modulov (*Image, ImageDraw* a *ImageFont*) sme z doteraz získaných dát vytvorili vlastný obrázok, resp. obrázky, s farebným notovým zápisom.

Veľkosť obrázka sme nastavili na 22 a šírku na 15 centimetrov. Následne tento rozmer prepočítali na pixely a vytvorili obrázok.

Podľa počtu nôt, ktoré sa majú vykresliť, sa vypočíta počet potrebných riadkov. Na jednu stranu (jeden obrázok) sa zmestia maximálne štyri riadky a na jeden riadok maximálne 10 zafarbených nôt. Každý riadok začína huslovým kľúčom.

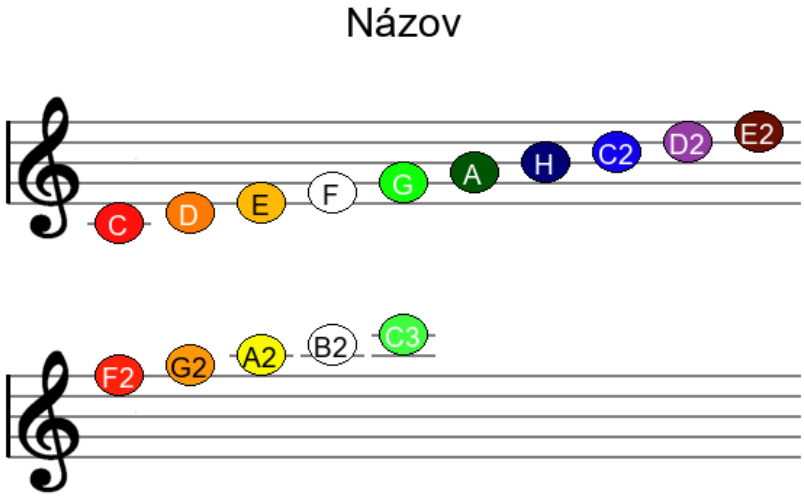
Pozíciu každej noty sme na základe vedomostí o notovom zápise vypočítali a aplikovali. Farby jednotlivých nôt zadávame ako parameter pre vykreslenie každej z nich - podľa farieb tónov zadaného xylofónu.

Tvar nôt sme zvolili farebné guličky, a to bez ohľadu na dĺžku pôvodnej noty. Takto sme sa rozhodli postupovať na základe cieľovej skupiny, ktorej sú tieto noty určené - a teda detom. Keď sme pracovali s knižnicami, ktoré ponúkali možnosť generovania farebmých nôt, stretli sme sa s problémom neprehľadnosti daného notového zápisu, a tak sme sa rozhodli vytvoriť si vlastný.

Každá strana je očíslovaná a na jej vrchu je názov skladby, ktorý si môže užívateľ sám zvoliť.

Rozsah nôt je od C4 po C6.

V obrázku s notovým zápisom je vyhradený dostatok priestoru pre užívateľa na dopísanie textu skladby. Takto môže užívateľ urobiť po vytlačení si farebných nôt, alebo predtým pomocou grafického editora. Vloženie textu a jeho priradenie k notovému zápisu nie je súčasťou tejto práce.



Obr. 3.14: Ukážka farebných nôt s maximálnym rozsahom

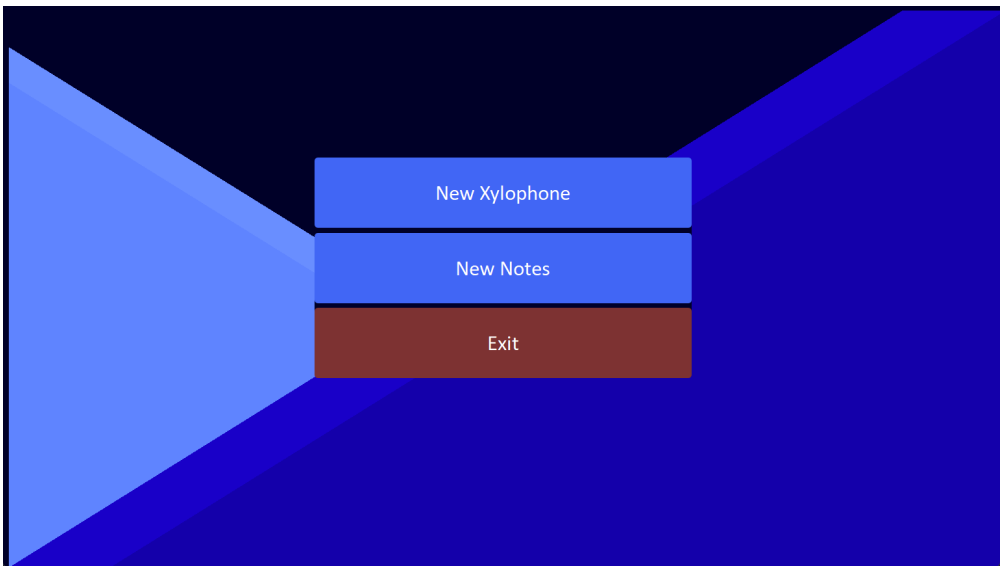
Kapitola 4

Užívateľské prostredie

Naša práca, ako sme už viackrát zmienili, sa zameriava na dve podstatné častí:

- spracovanie forografie xylofónu s cieľom extrakcie farieb tónov
- spracovanie notového zápisu v midi formáte a generovanie farebných nôt

preto sa aj naša aplikácia skladá z dvoch oddelených častí. Týmto sme docielili jednoduchosť ovládania aplikácie, a zároveň predišli nadväznosti chýb, ktoré by mohol úžívateľ pri práci zadať.



Obr. 4.1: Úvodná obrazovka aplikácie

Po spustení pythonovskej aplikácie sa zobrazí úvodná obrazovka, na ktorej si vieme vybrať z dvoch možností. Tlačidlom *New Xylophone* sa dostaneme do tej časti aplikácie, ktorá sa zaoberá spracovaním forografie a extrahovaním farieb tónov, tlačidlom *New Notes* do časti spracovania notového zápisu a generovania ferebných nôt.

4.1 Vytvorenie nového xylofónu

Po stlačení tlačidla *New Xylophone* na úvodnej obrazovke sa nám zobrazí obrazovka, v ktorej môžme pracovať dvomi následovnými spôsobmi:

1. Výber fotografie xylofónu a jej následné spracovanie
2. Vytvorenie si vlastného xylofónu

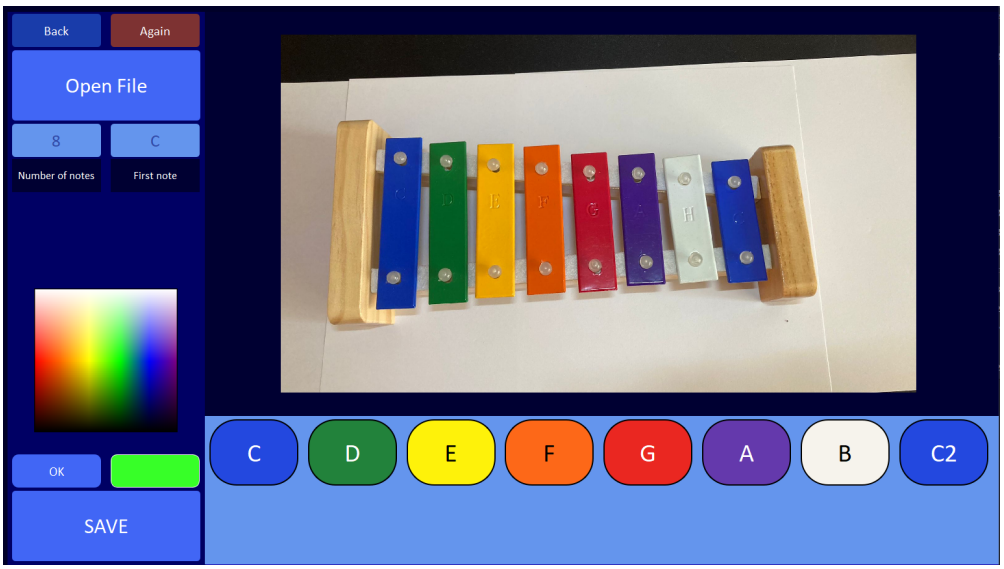
Pomocou tlačidla *Open File* prehľadávame súbory v počítači a vyberieme forografiu xylofónu, s ktorou chceme pracovať. Táto sa hneď zobrazí v aplikácii. V nasledujúcom kroku zadáme počet drievok, resp. tónov (*Number of Notes*) a tón, ktorým vieme označiť prvé drievko xylofónu (*First Note*).

Obr. 4.2: Označené rohy xylofónu v samostatnom okne

Obe majú prednastavené hodnoty na 8 a C, takže tento krok nie je povinný. Po stlačení tlačidla *Run* sa zobrazí okno s vybranou fotografiou, na ktorej užívateľ označí štyri rohy xylofónu a stalčí tlačidlo *Enter*. Po tomto sa spustí program, ktorý má zabezpečiť spracovanie zadaných dát a vrátiť extrahované farby xylofónu. Tieto farby sa zobrazia na spodnej časti aplikácie spolu s k nim priradenými tónmi.

Obr. 4.3: Farby extrahované z fotografie po spustení programu

Pre prípad, že by sa niektorá z fareb neextrahovala správnym spôsobom, alebo by chcel užívateľ ešte dodatočne upraviť niektorú z farieb, túto možnosť má po kliknutí na niektorý z farebných tónov. Následne sa zobrazí v ľavej časti farebná paleta, na ktorej si môže užívateľ vybrať farbu. Aktuálne vybranú farbu môžeme vidieť pod obrázkom s farebnou paletou. Vybranú farbu k tónu priradí až kliknutie na tlačidlo *OK*.

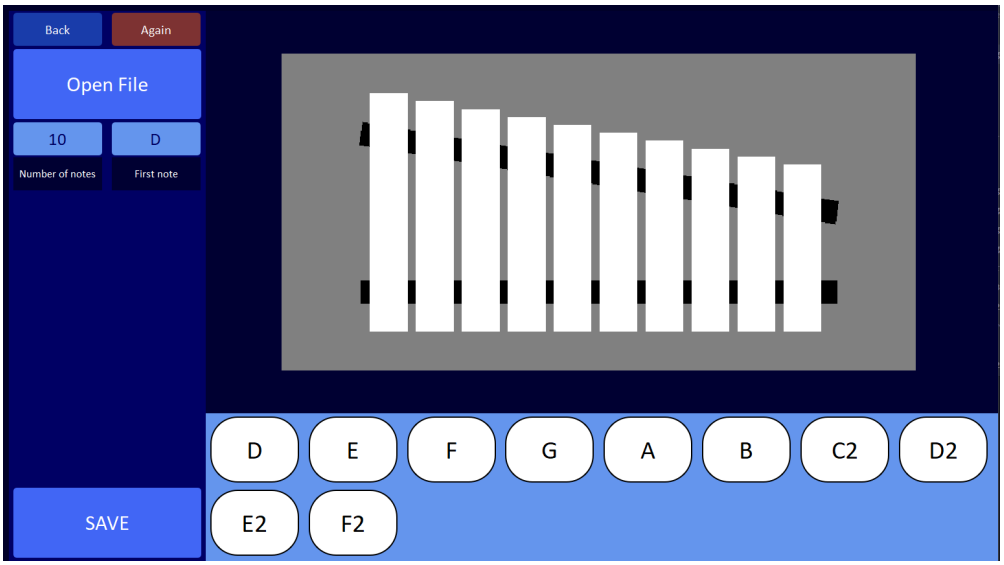


Obr. 4.4: Proces úpravy farby vybraného tónu

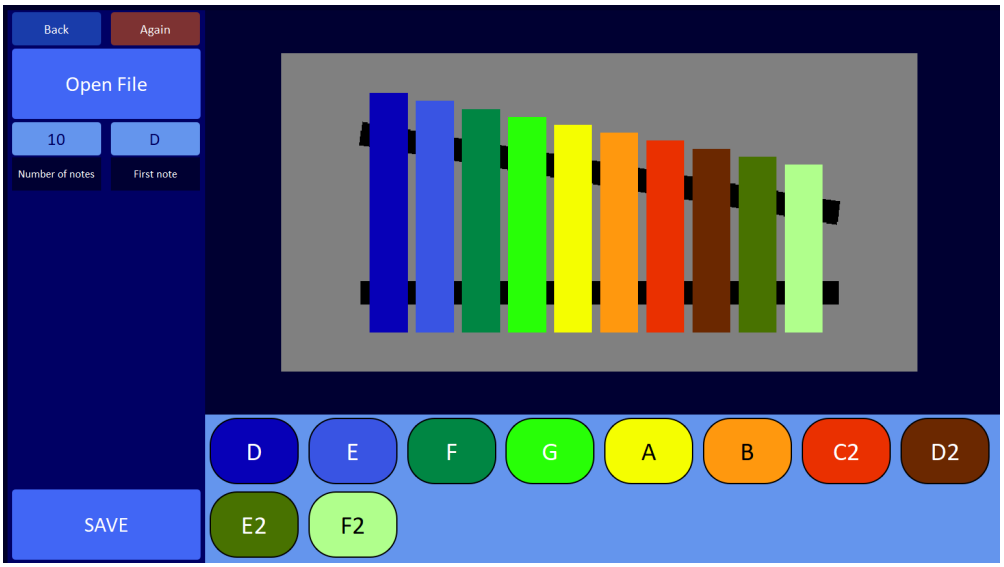
Druhou možnosťou, ktorú táto aplikácia ponúka, je vytvorenie si vlastného xylofónu. V tomto prípade si na začiatku zvolíme počet drievok xylofónu a prvú notu. Maximálny rozsah je 15 drievok - v tomto prípade musí ale xylofón začínať notou C, pretože maximálnu notu, ktorú vieme pri generovaní nôt získať na C3.

Po stlačení *Run* sa zobrazí obrázok nového xylofónu, ktorý bude obsahovať zadaný

počet bielych drievok. Tieto drievka vieme upravovať rovnakým spôsobom ako bolo vyššie spomínané.



Obr. 4.5: Novovytvorený xylofón pred zadaním farieb s rozsahom 10 drievok a počiatočným tónom D



Obr. 4.6: Xylofón po výbere farieb všetkých tónov

Ak sa užívateľ pomýlil v niektorom zo svojich krokov, môže stlačiť tlačidlo *Again* a začať proces nanovo.

Ak sa chce užívateľ vrátiť na úvodnú obrazovku, môže tak spraviť stlačením tlačidla *Back*.

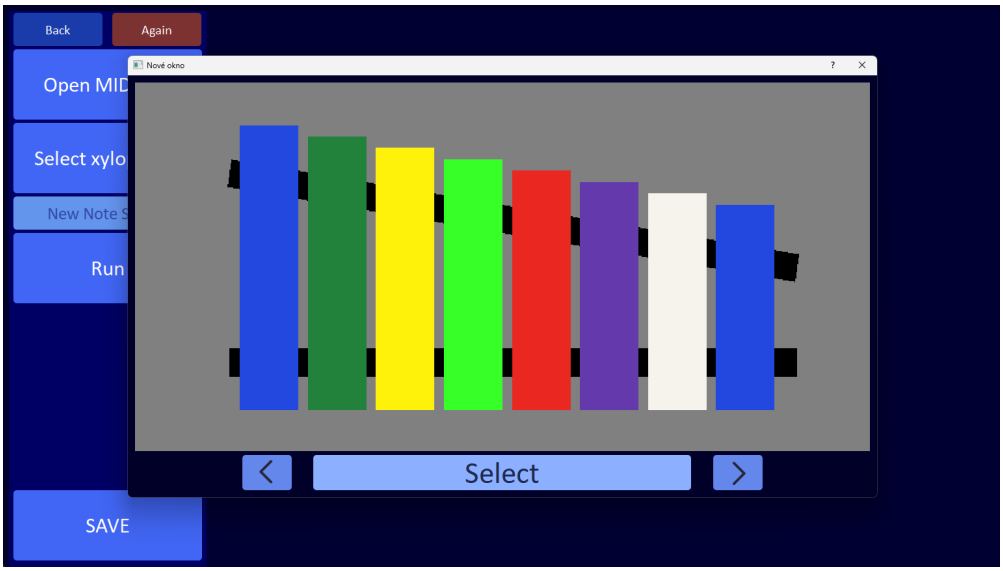
Keď je užívateľ spokojný so získanými, prípadne upravenými farbami, stlačí tlačilo *Save*, po ktorom sa vytvorený xylofón uloží medzi uložené xylofóny.

4.2 Vytvorenie farebných nôt

Ako sme už spomínali, druhou časťou aplikácie je spracovanie notového zápisu v midi formáte a generovanie farebných nôt. K tejto časti sa preklikneme stlačením tlačidla *New Notes*.

V tejto chvíli sa nám zobrazí plocha, v ktorej prebehne vytvorenie a zobrazenie výsledných nôt. V prvom kroku si po zakliknutí tlačidla *Open MIDI* otvoríme SMF súbor, z ktorého chceme čerpať informácie o notách pre vytvorenie farebných nôt. Druhou možnosťou je otvoriť textový súbor.

Následne si môžeme vybrať na získanie farieb jeden z xylofónov, ktoré sme si v minulosti vytvorili. Tento krok nie je povinný, pretože je predom nastavený výber xylofónu na ten, ktorý sme vytvorili naposledy.



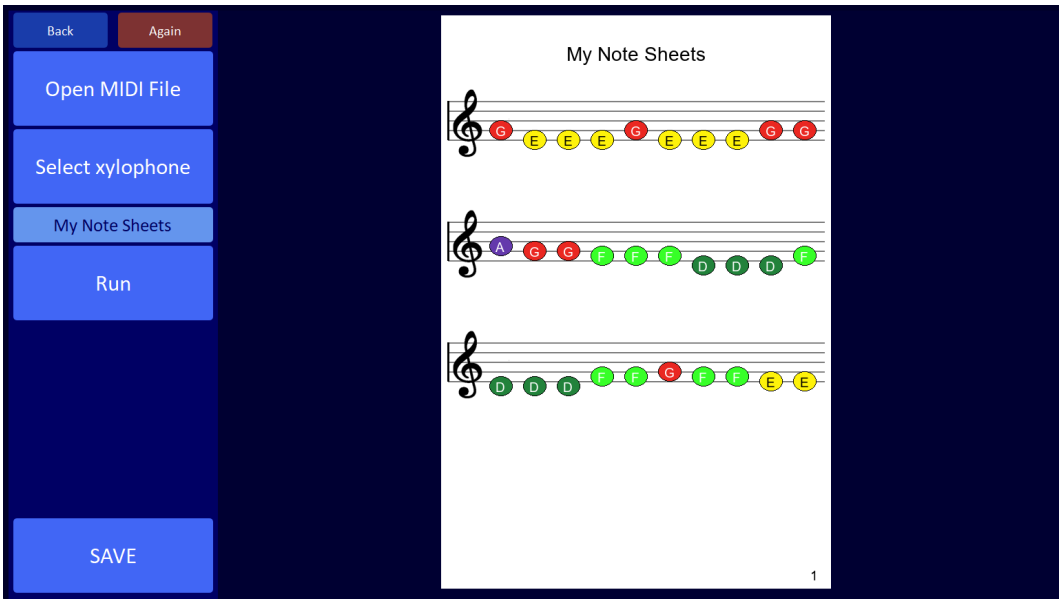
Obr. 4.7: Výber z uložených xylofónov pre vytvorenie farebných nôt

Potom už môžme len nastaviť názov skladby a stlačiť tlačidlo *Run*, čím sa vytvoria farebné noty a zobrazia sa v ploche.

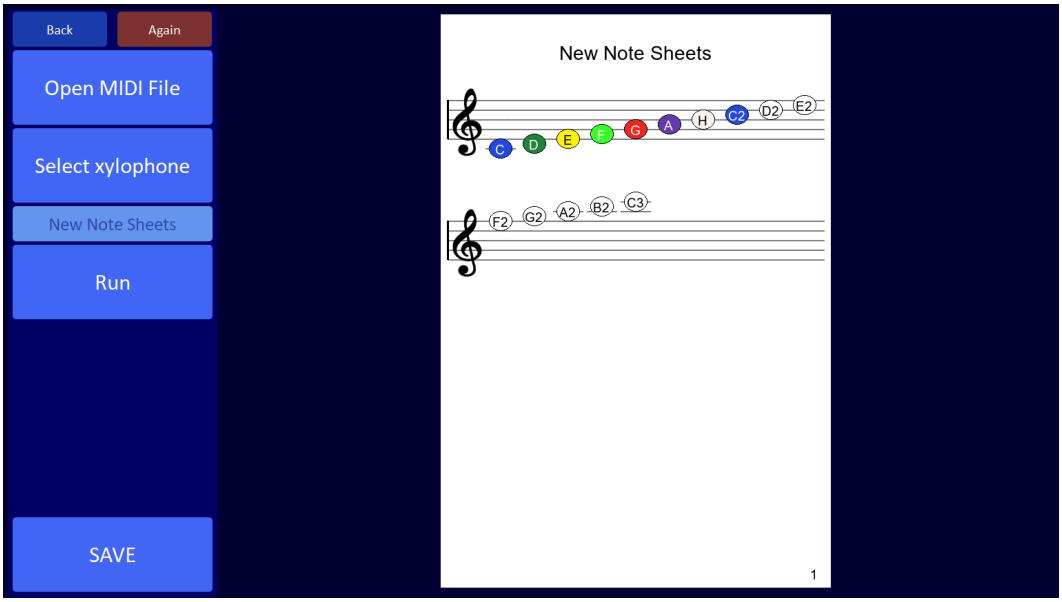
Aj po zobrazení vygenerovaných nôt môžeme tieto noty upravovať, a teda vybrať iný SMF súbor, iný z uložených xylofónov, či zmeniť názov nôt.

Ak sa noty nezmestili na jednu stranu, bude vytvorených viacero strán s notami medzi ktorými sa môžeme priamo v aplikácii preklikávať.

V prípade, že rozsah nôt načítaných zo súboru je väčší ako rozsah zvoleného xylofónu, pre takéto noty sa nastaví farba noty na bielu.



Obr. 4.8: Zobrazenie vytvorených nôt



Obr. 4.9: Zobrazenie vytvorených nôt presahujúcich rozsah xylofónu

Záver

Podarilo sa nám vytvorit' pythonovskú aplikáciu, ktorá dokáže spracovat' fotografiu xylofónu, extrahovat' z nej farby a priradit' ich jednotlivým tónom, a ktorá taktiež dokáže previest' notový zápis z midi formátu a vytvorit' obrázky s farebnými notami, ktoré zodpovedajú danému xylofónu.

Predpokladáme, že fotografia xylofónu je ostrá, dobre osvetlená, má jednofarebné pozadie bez vzorov a v najideálnejšom prípade obsahuje minimum tieňov. V prípade, že sa detekcia nepodarí, užívateľ môže farby editovat', a teda končený výsledok závisí na ňom.

Pri práci s obrázkom sme používali knižnicu *OpenCV*. Pred samotnou extrakciou farieb sme si obrázok najprv upravili. Pomocou perspektívnej transformácie sme eliminovali pozadie a rušivé aspekty fotografie. Následne po aplikácii kvantizácie farieb sme získali obrázok, ktorý obsahoval stanovený počet farieb (počet drievok xylofónu + 1 určené pre pozadie). Obrázok sme previedli na šedotónový, použili naň vyhladzujúci filter, adaptívne prahovanie a pomocou Houghovej transformácie získali množinu priamok v obrázku. Z nájdených priamok sme odstránili všetky tie, ktoré sme d'alej pri práci nepotrebovali a získali tak iba tie, ktoré ležia na hranách jednotlivých drievok. Vd'aka týmto priamkam sme už vedeli pre každé drievko lokalizovat' pixel, vd'aka ktorému sme získali jeho farbu a priradili ju danému tónu. Po zobrazení priradených farieb môže užívateľ ešte tieto farby upravovat'.

V druhej časti práce a aj samotnej aplikácie pracujeme s notovým zápisom vo formáte midi. Za použitia knižnice *music21* sme po načítaní a spracovaní SMF súboru vytvorili vlastné obrázky s farebnými notami, zodpovedajúcimi už spracovanému xylofónu. Maximálny rozsah pre noty je stanovený od C4 po C6. Pre užívateľa sme pridali možnost' načítat' textový súbor, ktorý obsahuje jednoduchý zápis nôt. Tento súbor program rovnako spracuje a vygeneruj farebné noty. Obrázky sme vytvorili pomocou knižnice *PIL*. Užívateľ môže počas úprav farebných nôt zmenit' výber SMF alebo textového súboru a aj svoj výber spomedzi uložených xylofónov.

Pre vytvorenie užívateľského prostredia sme použili knižnicu *PyQT*.

Aplikáciu sme otestovali na 15 unikátnych obrázkoch xylofónu. Na základe testovania sme vyhodnotili, že pravdepodobnost' správneho priradenia farieb na jeden odfotený xylofón je 87.33%. V buducnosti by sme určite chceli aplikáciu otestovat' na

rodičoch, ktorí majú o takto vytvorené farebné noty záujem.

Literatúra

[1] Opencv: About. https://opencv.org/about/.

[2] Opencv: Hough line transform. https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html.

[3] Opencv: Image thresholding. https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html.

[4] Opencv: K-means clustering in opencv. https://docs.opencv.org/3.4/d1/d5c/tutorial_py_kmeans_opencv.html.

[5] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. Ö'Reilly Media, Inc.", 2008.

[6] Antónia Droppová. Elementárna hudobná teória. *Prešov: Pedagogická fakulta Prešovskej univerzity v Prešove*, (s 104), 1998.

[7] Paul Heckbert. Color image quantization for frame buffer display. *ACM Siggraph Computer Graphics*, 16(3):297–307, 1982.

[8] Charles Loop and Zhengyou Zhang. Computing rectifying homographies for stereo vision. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages 125–131. IEEE, 1999.

[9] Sergio Márquez-de Silva, Edgardo Felipe-Riverón, and Luis Pastor Sánchez Fernández. A simple and effective method of color image quantization. In *Progress in Pattern Recognition, Image Analysis and Applications: 13th Iberoamerican Congress on Pattern Recognition, CIARP 2008, Havana, Cuba, September 9-12, 2008. Proceedings 13*, pages 749–757. Springer, 2008.

[10] Eleanor Selfridge-Field et al. *Beyond MIDI: the handbook of musical codes*. MIT press, 1997.

[11] Fayez Tarsha-Kurdi, Tania Landes, and Pierre Grussenmeyer. Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. In *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, volume 36, pages 407–412, 2007.

[12] Da-Chun Wu and Ming-Yao Chen. Information hiding in standard midi files based on velocity reference values. *Int. J. Netw. Secur.*, 18(2):274–282, 2016.