VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

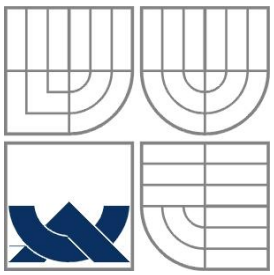
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ A PŘEHRÁVÁNÍ NOT Z FOTOGRAFIE

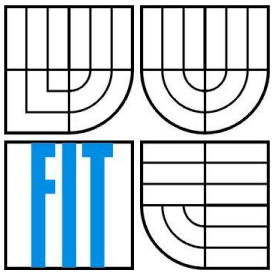
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. JIŘÍ STANĚK
AUTHOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ A PŘEHRÁVÁNÍ NOT Z FOTOGRAFIE

SHEET MUSIC RECOGNITION AND PLAYING FROM PHOTOGRAPHY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. JIŘÍ STANĚK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ALENA PAVELKOVÁ

BRNO 2015

Abstrakt

Tato práce se zabývá vývojem aplikace pro automatické rozpoznávání not. Aplikace je určena pro mobilní telefony s operačním systémem Android. Práce obsahuje stručný úvod do problematiky a uvádí některá existující řešení problému. Jsou zde popsány použité metody pro zpracování obrazu a klasifikaci. Dále je popsán návrh a implementace samotné aplikace, kde je uveden způsob detekce a odstranění notových linek, detekce a zpracování hudebních symbolů a jejich klasifikace. Následuje vyhodnocení finální aplikace a shrnutí dosažených výsledků.

Abstract

This work deals with development of an application for optical music recognition. This application is designed for mobile phones with Android operating system. The work includes a brief introduction to the problem and introduces some existing solutions. There are described methods for image processing and classification, which are used in final application. It also shows the design and implemenation of the final aplication, where used methods for detection and removal of staff lines, detection and processing of musical symbols and their classification are described. The evaluation of the final application and a summary of achieved results are shown in the end of this work.

Klíčová slova

zpracování obrazu, OMR, notace, LIBSVM, OpenCV, odstranění notových linek, Android, hudba

Keywords

image processing, OMR, notation, LIBSVM, OpenCV, staff lines removal, Android, music

Citace

Staněk Jiří: Rozpoznávání a přehrávání not z fotografie, diplomová práce,Brno, FIT VUT v Brně, 2015

Rozpoznávání a přehrávání not z fotografie

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Aleny Pavelkové. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Staněk
27. května 2015

Poděkování

Rád bych poděkoval vedoucí práce Ing. Aleně Pavelkové za vstřícný přístup a cenné rady poskytnuté při konzultacích.

Obsah

1 Úvod

S rozvojem komunikačních technologií vzrůstá potřeba digitalizace dokumentů, což umožňuje s dokumenty pracovat v elektronické podobě (ukládat je, editovat). V posledních několika desítkách let se lidé zabývají digitalizací notových zápisů, neboli automatickým rozpoznáváním not (optical music recognition, dále OMR). Hlavním cílem OMR je digitalizace hudby a její elektronické přehrávání.

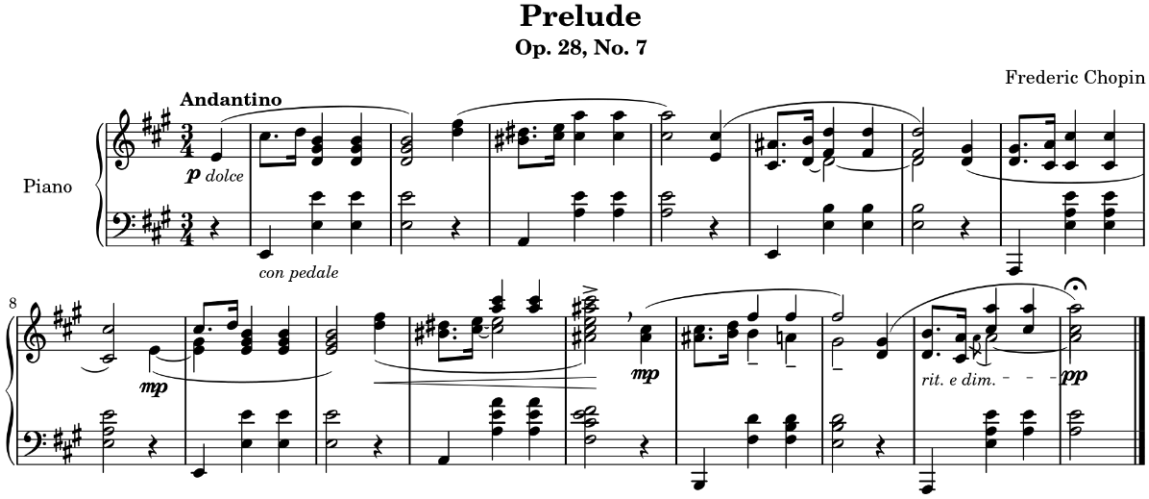
Cílem této práce je takovou aplikaci vytvořit. Tato aplikace je určena pro mobilní telefony s operačním systémem Android. To přináší řadu problémů. Jedním z nich je, že vstupní obraz bude pořízen pomocí fotoaparátu. Fotoaparáty mobilních telefonů nedosahují kvalit digitálních fotoaparátů, či skenerů a je potřeba se vypořádat s různými nedostatky, především geometrickou deformací obrazu. Dalším problémem může být výpočetní výkon mobilních telefonů, který stále velmi zaostává za moderními stolními počítači, tudíž se zde naskýtá otázka zpracování v reálném čase.

Práce je rozdělena do několika kapitol. V kapitole 2 je popsán způsob zápisu hudby do notové osnovy, jsou zde uvedeny jednotlivé hudební symboly a jejich význam. Kapitola 3 prezentuje existující řešení problému. OMR je zde rozděleno do několika podproblémů, u nichž jsou popsána řešení různých autorů. Jsou zde zmíněna řešení starší, která většinou počítají s ideálním vstupem, i současná řešení, snažící se o co největší robustnost. V kapitole 4 je uveden princip metod pro zpracování obrazu a klasifikaci použitých ve výsledné aplikaci. Kapitola 5 pojednává o výsledné aplikaci. Je zde popsán návrh a implementace aplikace. Jsou zde zmíněny problémy, které se během implementace vyskytly a případně jejich řešení. V kapitole 6 je popsán způsob testování a vyhodnocení aplikace a jsou zde ukázány výsledky důležitých částí, kterými jsou klasifikace a doba zpracování obrazu. Závěr shrnuje dosažené výsledky a pojednává o nedostatcích aplikace a o možnostech budoucího vývoje.

2 Notace

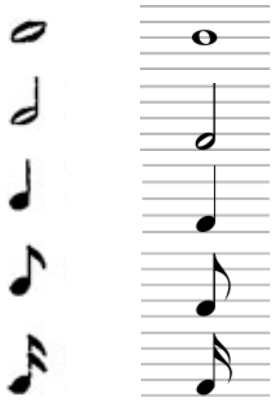
Před začátkem tvorby programu pro automatické rozpoznávání not, je třeba znát jednotlivé hudební symboly a způsob, jakým se z těchto symbolů vytváří notový zápis. Pouze s touto znalostí lze vytvořit kvalitní a robustní program.

V historii různé národy v různých dobách používaly odlišné způsoby zápisu hudby. V dnešní době se ustálilo užívání moderního notového zápisu (Obrázek 2.1).



Obrázek 2.1: Příklad moderního notového zápisu.
Převzato z: http://en.wikipedia.org/wiki/Musical_notation

Pro zápis hudebních symbolů neexistuje jediný font, proto mohou stejné symboly v různých notových zápisech vypadat odlišně (Obrázek 2.2), což komplikuje jejich rozpoznávání.

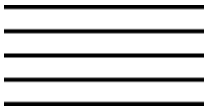


Obrázek 2.2: Příklad odlišných fontů pro hudební symboly.
Převzato z: Vlevo [22], vpravo http://en.wikipedia.org/wiki/List_of_musical_symbols

2.1 Čáry pro notový zápis

V notovém zápisu se vyskytují především dva typy čar, čáry vodorovné (linky) a čáry svislé. Vodorovné linky udávají výšku not a svislé čáry dělí skupiny not do sekcí.

- **Notová osnova**
  Notová osnova se skládá z pěti vodorovných rovnoběžných linek a šesti mezer (Obrázek 2.3). Linky jsou označeny jako první, druhá, třetí, čtvrtá a pátá, počínaje nejspodnější. Mezery jsou značeny podobně, počínaje mezerou pod (mezera pod první linkou), následuje první mezera, druhá, třetí, čtvrtá a mezera nad (nad pátou linkou) [22].

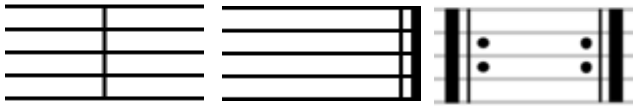


Obrázek 2.3: Notová osnova.
Převzato z: http://en.wikipedia.org/wiki/List_of_musical_symbols

- **Pomocné linky**
  Pomocné linky jsou krátké čáry, sloužící k rozšíření notové osnovy ve vertikálním směru (Obrázek 2.4). Noty mohou být zápsány na tyto čáry nebo pod/nad ně [22].

Obrázek 2.4: Pomocné čáry.
Převzato z: http://en.wikipedia.org/wiki/List_of_musical_symbols

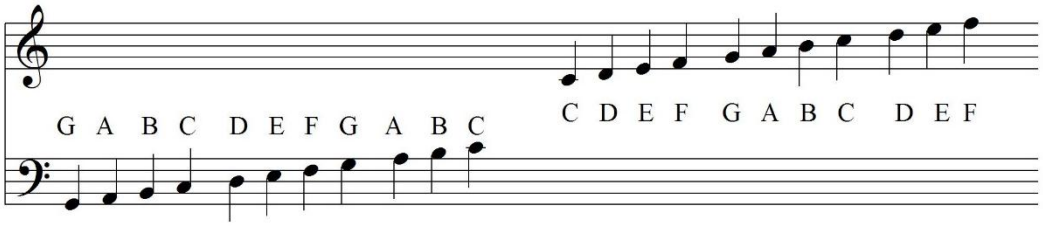
- **Taktová čára**
  Taktová čára je svislá čára protínající notovou osnovu a dělící ji na takty. Takty dělí noty do skupin o stejném počtu dob (součet délek not ve skupině). Taktová čára může být dvojitá, což označuje konec větší sekce (sloka, či celá skladba). Přidá-li se k dvojité taktové čáře dvojice teček, vzniklý symbol značí sekci, která má být opakována [22]. Obrázek 2.5 vyobrazuje různé druhy taktových čar.



Obrázek 2.5: Druhy taktových čar.
Převzato z: http://en.wikipedia.org/wiki/List_of_musical_symbols.

2.2 Notové klíče

Klíč je symbol určující jakým tónům (výškám) odpovídají jednotlivé linky a mezery v notové osnově. Klíčů je celá řada, nejpoužívanější jsou G klíč, neboli houslový klíč, a F klíč, neboli basový klíč (Obrázek 2.6). Notová osnova se poté nazýva houslová osnova nebo basová osnova [22].

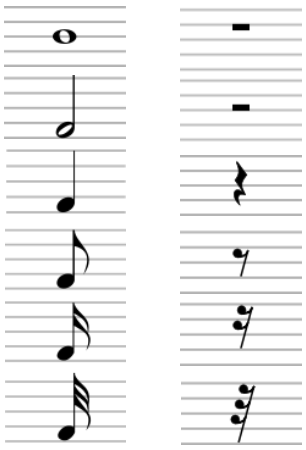


Obrázek 2.6: Notové klíče.
Převzato z http://www.infobarrel.com/Media/The_Grand_Staff.

2.3 Noty a pomlky

- **Noty**
  Nota je symbol označující dobu trvání. Pokud je umístěna v notové osnově, označuje, že jistý tón má znít po určitou dobu (Obrázek 2.7 vlevo).

  Nota se skládá z jedné, dvou, nebo tří částí, nazývaných hlava, noha a háček (nebo ocas, či prapor). Noha, která směřuje vzhůru, se nachází na pravépravé straně hlavy, v opačném případě na levé straně hlavy. Háček je vždy na pravé straně nohy. Noha směřuje vzhůru, pokud je hlava noty pod třetí čarou, pokud je nad třetí čarou, noha směřuje dolů [22].



Obrázek 2.7: Noty a pomlky. Odshora: nota celá, půlová, čtvťová, osminová, šestnáctinová, dvaatřicetinová.
Převzato z: http://en.wikipedia.org/wiki/List_of_musical_symbols.

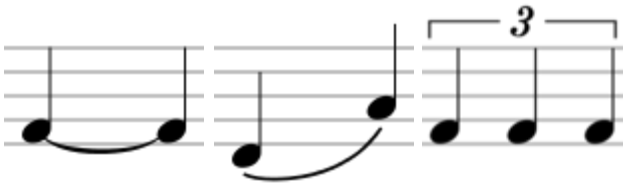
Noty malých hodnot (osminová a menší) bývají často spojovány do skupin zvaných složené noty. Nohy not jsou pak spojeny jednou (či více – záleží na typu noty) tlustou čarou.

Za notou může následovat tečka, která zvyšuje hodnotu noty o polovinu. Tečka se vždy zapisuje do mezery. Obrázek 2.1 obsahuje složené noty a noty s tečkou.

- **Pomlky**
  Pomlka je znak signalizující ticho po určitou dobu. Pomlky vyobrazuje Obrázek 2.7 vpravo.

- **Vztahy mezi notami**
  Existuje celá řada vztahů mezi notami, nejznámějšími jsou následující vztahy popsané Gehrkensem [22] (Obrázek 2.8):

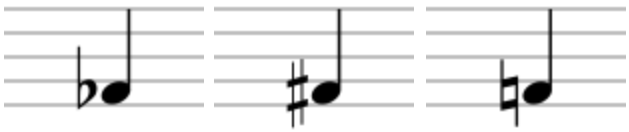
- **Tie**
  Tie je křivka spojující dvě (nebo více) stejné noty (stejný typ a výška). Určuje, že noty mají být zahrány jako jedna nota o délce rovnající se součtu délek obou not.
- **Slur**
  Slur má stejný význam jako tie, ale může spojovat různé noty. Může mít i další významy popsány např. Gehrkensem [22], které pro tuto práci nejsou důležité.
- **Triplet**
  Triplet (bývá nejčastější, jsou i jiné typy jako duplet, quadruplet,...) označuje skupinu tří not, které mají být zahrány s dobou běžně danou pouze dvěma notami tohoto typu.



Obrázek 2.8: Vztahy mezi notami. Vlevo tie, uprostřed slur, vpravo triplet.
Převzato z: http://en.wikipedia.org/wiki/List_of_musical_symbols

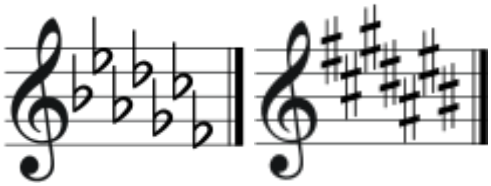
2.4 Posuvky a předznamenání

- **Posuvky**
  Posuvky mění výšku not, které za nimi bezprostředně následují. Efekt posuvek platí v celém taktu. Nejčastější jsou tři typy popsané Gehrkensem [22] (Obrázek 2.9):
  - **Béčko**
    Béčko je symbol snižující výšku noty o polovinu.
  - **Křížek**
    Křížek je symbol zvyšující výšku noty o polovinu.
  - **Odrážka**
    Odrážka ruší předchozí posuvky v dané výšce.



Obrázek 2.9: Posuvky. Vpravo béčko, uprostřed křížek, vlevo odrážka.
Převzato z: http://en.wikipedia.org/wiki/List_of_musical_symbols

- **Předznamenání**
  Předznamenání je skupina posuvků (béčka nebo křížky) vyskytující se na začátku osnovy za notovým klíčem (Obrázek 2.10). Tyto posuvky ovlivňují celou notovou osnovu, tj. zvyšují (či snižují) výšku not (se stejnou výškou jako posuvka) v celé notové osnově.



Obrázek 2.10: Předznamenání.
Převzato z: http://en.wikipedia.org/wiki/List_of_musical_symbols.

2.5 Časové značky

Časové značky udávájí počet dob v každém taktu. Dělí se na dva typy popsané Gehrkensem [22]:

- **Jednoduché časové značky**
  Tyto značky se zapisují pomocí dvou čísel zapsaných nad sebou. Zapisují se za předznamenání. Spodní číslo udává typ noty a horní číslo udává počet těchto not v jednom taktu. Např. Obrázek 2.11 vlevo zobrazuje tří čtvrtinový čas, což znamená, že každý takt má délku odpovídající délce třem čtvrťovým notám. Existuje speciální značka pro čtyř čtvrťový čas (označovaný jako běžný čas) (Obrázek 2.11 vpravo).
- **Složené časové značky**
  Tyto značky jsou složeny z více jednoduchých časových značek. Pro tuto práci nejsou důležité.

Obrázek 2.11: Jednoduché časové značky. Vlevo značka složená ze dvou čísel,
vpravo značka pro běžný čas.
Převzato z: http://en.wikipedia.org/wiki/List_of_musical_symbols

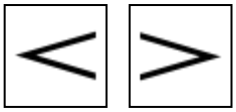
2.6 Dynamika

Dynamika udává hlasitost hrané hudby. Nějčastější jsou následující typy popsané Gehrkensem [22]:

- **Pianisissimo** (*ppp*) – tak potichu, jak je to jen možné
- **Pianissimo** (*pp*) – velmi potichu
- **Piano** (*p*) – potichu
- **Mezzo piano** (*mp*) – středně potichu
- **Mezzo forte** (*mf*) – středně hlasitě
- **Forte** (*f*) – hlasitě
- **Fortissimo** (*ff*) – velmi hlasitě
- **Fortisissimo** (*fff*) – tak hlasitě, jak je to jen možné

Další dva typy značek, které popsal Gehrkens [22] udávají postupnou změnu hlasitosti (Obrázek 2.12):

- **Crescendo** – postupně zvyšování hlasitosti
- **Decrescendo** – postupné snižování hlasitosti



Obrázek 2.12: Dynamické značky. Vlevo crescendo, vpravo decrescendo.
Převzato z: http://en.wikipedia.org/wiki/List_of_musical_symbols

3 Existující řešení automatického rozpoznávání not

Automatické rozpoznávání not (dále jen OMR – Optical Music Recognition) z obrazu je složitý komplexní problém, kterým se zabývá řada lidí již několik desítek let. Tento problém je velice odlišný od optického rozpoznávání textu (OCR – Optical Character Recognition), tudíž běžné metody užívané pro OCR zde nelze použít. Vyplývá to ze způsobu, jakým jsou noty zapisovány. Při pohledu na notový zápis, lze totiž vidět, že jednotlivé hudební symboly od sebe nejsou odděleny, nýbrž jsou propojeny vodorovnými čarami, tvořícími notovou osnovu. Aby ovšem bylo možné zpracovat notový zápis, je nutné jednotlivé symboly izolovat. Jak lze ale izolovat symboly, které jsou spolu propojené? Tato otázka je zásádní pro celé OMR a proto tento problém bude dále podrobněji rozebrán a budou ukázány různé přístupy k jeho řešení.

Jsou-li získány izolované symboly, dalším krokem je jejich roztřídění do skupin podobných symbolů. Častým řešením je rozdělit symboly na vertikální a ostatní, kde vertikální symboly jsou většinou noty a taktové čáry.

Následuje klasifikace získaných symbolů. Zde je určeno, co jednotlivé symboly představují (půlová nota, posuvka a její typ, atd.) a také kde v osnově se nacházejí, což určuje jejich výšku.

Posledním krokem je zahrnutí hudebních pravidel. V tomto kroku je potřeba zkontrolovat, zda se podařilo veškeré symboly správně klasifikovat. Probíhá zde převážně kontrola počtu dob v jednotlivých taktech a následná korekce délky not. OMR lze tedy rozdělit do následujících podproblémů:

- Předzpracování dat
- Segmentace
- Klasifikace
- Post-processing

V následujících kapitolách budou probrány jednotlivé kroky a budou zmíněny existující přístupy k jejich řešení.

3.1 Předzpracování vstupních dat

V této podkapitole budou probrány metody pro řešení nejdůležitějšího kroku OMR, tj. izolování jednotlivých objektů.Nejprve ale bude popsáno zpracování vstupních dat.

3.1.1 Vstupní data

Na vstupu OMR je většinou obraz pořízený pomocí skeneru nebo fotoaparátu. Volí se rozlišení většinou 100 – 400 dpi, které je dostatečné a zároveň nevyžaduje příliš velký výpočetní výkon.

3.1.2 Prahování obrazu

Vstupní obraz je nejčastěji šedotónový a je potřeba jej binarizovat. K tomu slouží adaptivní prahování (4.3.1).

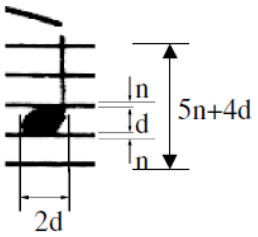
3.1.3 Rotace obrazu

Málokdy se povede pořídit obraz dokonale rovně a proto musí být natočen tak, aby byly notové osnovy co nejvíce vodorovné, což usnadní jejich detekci. Existuje více přístupů k narovnání obrazu. Jedním z těchto přístupů, užitým Fornésem [10], je aplikace Houghovy transformace pro nalezení notových linek a vypočítání úhlu rotace. Jiným způsobem, který popsal Capela [5], je aplikace horizontální projekce (4.5) v různých úhlech a ponechání si výsledku s největšími lokálními maximy. Dalším zajímavým způsobem užitým Fujinagou [13] je průchod obrázku po sloupcích a hledání prvního černého pixelu v každém sloupci. Mezi po sobě jdoucími pixely se počítájí úhly a obrázek je natočen pomocí nejčastěji se vyskytujícího úhlu.

3.1.4 Detekce notových linek

Před detekcí jednotlivých symbolů, je potřeba zjistit údaje o notovém zápisu. Důležité jsou dvě míry: průměrná tloušťka notové linky (n) a průměrná vzdálenost notových linek v osnově (d). Všechny OMR systémy začínají tímto krokem [1].

Získané dva údaje značně usnadní detekci jednotlivých hudebních symbolů, jelikož jejich rozměry na nich silně závisí (Obrázek 3.1). Toto značení bude použito v následujícím textu.



Obrázek 3.1: Rozměry hlavy noty.
Převzato z [2].

Pro získání tloušťky linek a jejich vzájemné vzdálenosti, je potřeba nejprve linky detekovat. Pro detekci linek existuje mnoho způsobů, z nichž některé budou popsány dále.

Jsou-li detekovány linky a získány potřebné údaje, je nutné se rozhodnout, jaký bude další postup. Možné jsou dva způsoby.

Linky lze v obraze ponechat a pracovat i s nimi. Toto řešení značně ztěžuje detekci hudebních symbolů, jelikož, jak již bylo řečeno, ty jsou notovými linkami propojeny a nejsou izolovány. Výhodou tohoto řešení je, že symboly nejsou poškozeny, což se při odstraňování linek může snadno stát.

Druhým způsobem je odstranění notových linek. Výhodou je, že poté nám zůstanou izolované symboly, které lze snadno detekovat. Odstranění notových linek ovšem musí probíhat velice opatrně, jelikož zde může dojít k poškození symbolů a poté znesnadnění jejich detekce a klasifikace.

Většina řešení se přiklání ke druhému způsobu, tj. odstranění notových linek. Proto zde vzniklo velké množství různých návrhů, jak docílit co nejlepšího výsledku.

Následuje popis několik způsobů detekce notových linek a jejich odstranění (či ponechání).

Ponechání notových linek

Bellini, Bruno a Nesi [2] se rozhodli notové linky neodstraňovat. Nejprve pomocí RLC (4.6) určí průměrnou šířku linky a průměrnou vzdálenost mezi linkami. Dále detekují jednotlivé notové osnovy pomocí horizontální projekce v úzkém posuvném okně, na základě pěti stejně vzálených řádků. Způsob, jakým v získaných notových osnovách detekují objekty je popsán v (3.2.1).

Odstranění notových linek

Jak bylo zmíněno, většina řešení se přiklání k odstranění notových linek před detekcí samotných objektů. Dá se říct, že je to nejdůležitější krok celého OMR a na tom, jak dobře se podaří linky odstranit, závisí úspěšnost dalších kroků (segmentace a klasifikace).

Od prvních pokusů o vyřešení tohoto problému až do současnosti vzniklo mnoho metod pro detekci a odstranění notových linek. Nejstarší z nich (také nejjednodušší) spoléhají na kvalitní vstupní obraz, jiné, novější, se snaží o co největší robustnost a poradí si i s perspektivní deformací, či pokřivenými linkami. Následuje popis několika vybraných metod.

- **Capela A. a spol. [4][5]**
  Capela a spol. nahlíží na obrázek jako na graf, čáry jsou spojité cesty mezi dvěma okraji obrázku. Notová linka je poté nejkratší cesta spojující levý a pravý okraj. Graf je uspořádán tak, že v uzlech jsou jednotlivé pixely a hrany spojují sousední pixely. Hrany mají váhu danou intenzitou pixelu. Poté při průchodu spojité cesty je počítána její cena, což je součet vah na hranách. Notová linka je cesta s minimální cenou. Postupují tak, že vždy najdou nejkratší cestu (označující notovou linku) a hned ji odstraní. Odstraňují vertikální úseky pixelů o velikosti větší než $2n$. Hledání cest může skončit dvěma způsoby:
  1. Poslední nalezená cesta obsahuje malé množství černých pixelů v porovnání s první nalezenou cestou.
  2. Tvar poslední nalezené cesty se hodně liší od první nalezené cesty. Zde kontrolují rozdíly ve vertikálním směru.

- **Rossant F. [16]**
  Rossant lokalizuje jednotlivé notové linky pomocí horizontální projekce a spočítá průměrnou vzdálenost dvou linek. Každá linka je určena jednou y-ovou souřadnicí. Poté prochází každou linku v horizontálním směru a počítá velikost vertikálního segmentu černých pixelů nad a pod linkou. Pokud velikost tohoto segmentu je menší než typická šířka linky (čtvrtina průměrné vzdálenosti dvou linek), je tento segment odstraněn. Obrázek 3.2 zobrazuje výsledek této metody.

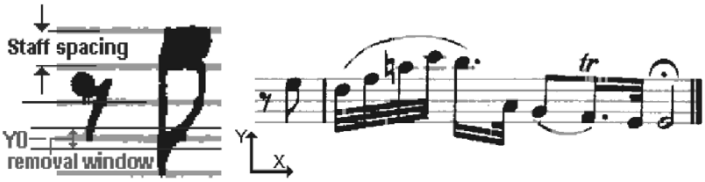
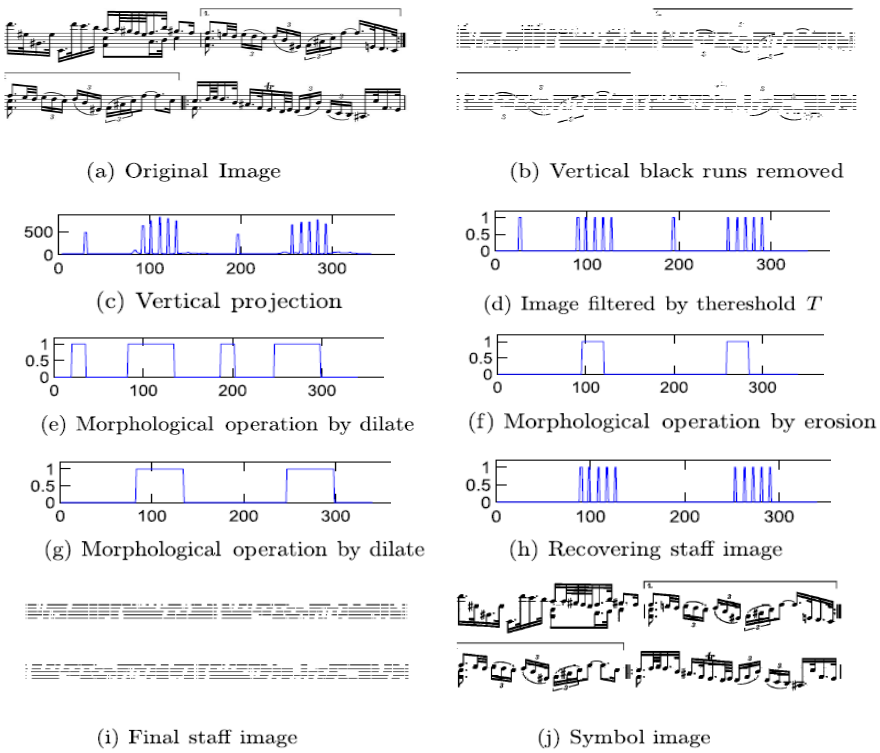


Fig. 2. Staff lines removal (removed pixels in light gray).

Obrázek 3.2: Rossant F.: metoda odstranění notových linek
(odstraněné pixely jsou šedé).
Převzato z [16].

- **Tajeripour F., Sotoodeh M. [21]**
  Tato metoda se zaměřuje na využití morfologických operací. Nejprve pomocí RLC zjistí průměrnou šířku linky a průměrnou velikost mezery mezi linkami. Poté jsou odstraněny všechny vertikální segmenty černých pixelů, které jsou větší než $2n$. Po tomto kroku zůstanou v obraze některé nechtěné objekty (dlouhé objekty, dynamické značky, text, ...). Tyto objekty je potřeba odstranit. Následuje výpočet vertikální projekce a odstranění všeho, co je menší než nějaký určený práh. Poté je aplikována dilatace na vzniklou projekci, čímž je dosaženo spojení vrcholů čar patřících do stejné osnovy. Následuje morfologické otevření, díky čemuž zůstanou ve výsledné projekci pouze notové osnovy. Strukturním elementem morfologických operací je vektor o velikosti $3d$. Obrázek 3.3 vyobrazuje postup této metody.



Obrázek 3.3: Tajeripour F., Sotoodeh M.: metoda odstranění notových linek.
Převzato z [21].

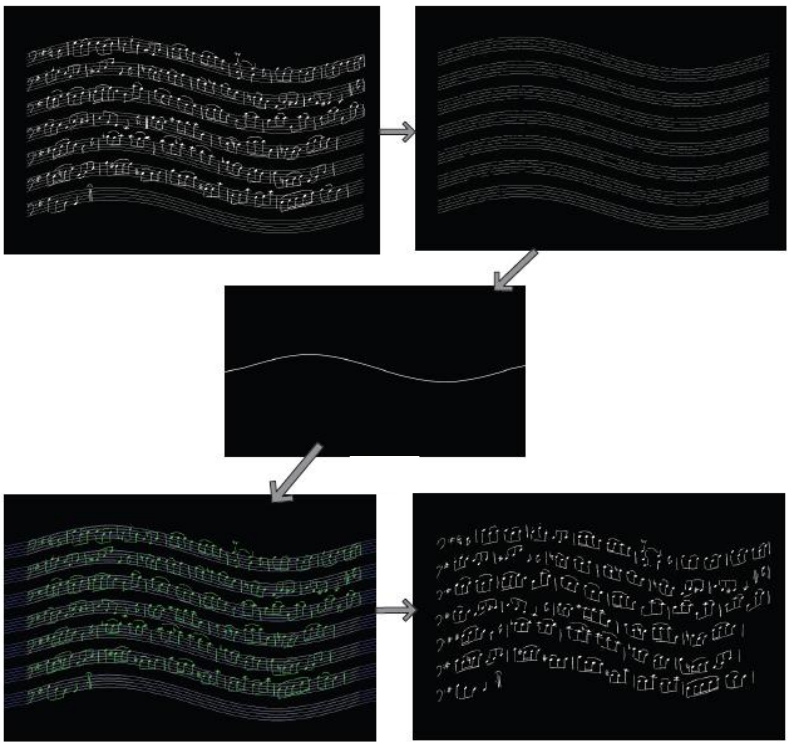
- **Lu S. a spol. [15]**
  Další metodou, snažící se o co největší robustnost, je metoda zakladájící se na tvorbě modelu tvaru notové linky a jeho aplikaci při detekci notových linek. Podle autorů je tato metoda

schopná se vypořádat s různými deformacemi notových osnov (zakřivení, rotace, necelistvost notových linek, různá šířka linek, atd.).

Postup metody zobrazuje Obrázek 3.4. Prvním krokem je výpočet průměrné tloušťky linky a průměrné vzdálenosti linek pomocí RLC a hrubé odstranění všeho, kromě notových linek na základě velikosti vertikálních segmentů černých pixelů, kdy jsou odstraněny všechny segmenty větší než práh T, kde $T = MIN\{2n, n + d\}$. Následuje tvorba modelu tvaru notové linky na základě zkoumání orientace pixelů. Obraz je procházen po sloupcích a zkoumá se změna pixelů, odpovídajících stejné notové lince, ve vertikálním směru. Pro lepší přesnost není brána změnu směru v sousedních sloupcích, ale vždy například po pěti sloupcích. Metoda počítá s tím, že jsou všechny linky paralelní a počítá model pro jednu linku. Po vytvoření modelu následuje rozdělení pixelů do skupin pomocí následujících tří kroků:

1. Pokud pixel není zařazen do žádné skupiny, je vytvořena skupina nová s tímto pixelem.
2. Je vytvořena křivka, která prochází tímto pixelem a má tvar stejný jako model tvaru notové linky.
3. Všechny pixely na této křivce spadají do stejné skupiny.

Dále jsou detekovány notové linky na základě počtu pixelů ve skupinách. Poté dojde k odstranění detekovaných linek opět pomocí velikosti vertikálního segmentu pixelů na základě výše zmíněného prahu T. Může se ovšem stát, že předpokládaná notová linka přesně neodpovídá skutečné notové lince. V tom případě se hledá nejbližší segment pixelů ve stejném sloupci.



Obrázek 3.4: Lu S. a spol.: metoda odstranění notových linek.
Převzato z [15].

3.2 Segmentace

Segmentace je další důležitou částí OMR. Její úspěšnost silně závisí na tom, jak dobře se povedlo odstranit notové linky (pokud byly odstraněny). Cílem segmentace je detekovat jednotlivé hudební symboly a zpracovat je, což většinou zahrnuje jejich rozdělení na základní prvky. To probíhá hlavně u not, které jsou rozděleny na hlavu a nohu a tyto části jsou klasifikovány zvlášť. Dále budou popsány existující metody pro segmentaci.

3.2.1 Notové linky ponechány

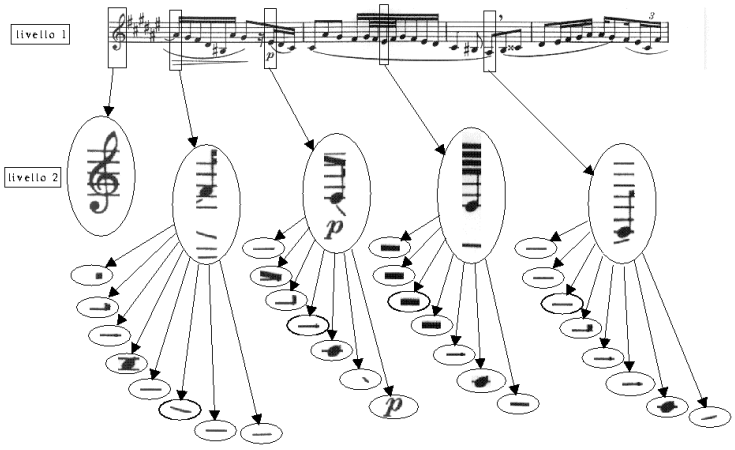
Jak bylo zmíněno, Bellini, Bruno a Nesi [2] se rozhodli pro ponechání notových linek a přistoupili rovnou k segmentaci, kterou provádí pomocí hierarchické dekompozice obrazu.Tento proces probíhá ve třech úrovních (Obrázek 3.5):

1. **Detekce notových osnov**
   Notové osnovy detekují tak, že si vezmou jen kousek obrázku (pár sloupců) a tento kousek obrázku zkoumají pomocí horizontální projekce, kdy hledají pěti čárové vzory reprezentující notovou osnovu.

2. **Dělení na symboly**
   Dalším krokem je rozdělit notovou osnovu na jednotlivé symboly. To probíhá ve třech krocích:

   - **Detekce složených not a izolovaných symbolů**
     V tomto kroku jsou hledány vertikální úseky na notové osnově, které neobsahují žádný symbol. Toho je docíleno pomocí posuvného okna širokého 3-4 px, které je posouváno zleva doprava a pomocí horizontální projekce je zkoumáno, zda okno obsahuje jen notové linky, nebo nějaký symbol.
   - **Detekce notových hlav**
     Do tohoto kroku vstupují objekty nalezené v předchozím kroku o šířce alespoň $2d$, což je šířka notové hlavy. Pomocí vertikální projekce jsou nalezeny notové hlavy a úsek je rozsekán na jednotlivé noty.
   - **Detekce ostatních symbolů**
     Nyní jsou zpracovány úseky vynechané předchozími kroky. Většinou se jedná o úseky obsahující symboly, které jsou velmi blízko u sebe (předznamenání). Izolování těchto symbolů probíhá pomocí hledání lokálních minim ve výsledku vertikální projekce.

3. **Dekompozice symbolů na základní prvky**
   Tento krok je rozdělen na dvě části pro symboly obsahující notovou hlavu a symboly, které ji neobsahují. Obě části jsou založeny na horizontální a vertikální projekci. Nejprve jsou odstraněny notové linky (jejich pozice je známa z kroku 2), poté jsou odstraněny nízkofrekvenční komponenty, tj. notová noha. Tím dojde k rozdělení noty na části, reprezentující hlavu a háček a ty jsou poté klasifikovány.



Obrázek 3.5: Bellini, Bruno a Nesi: segmentace notové osnovy.
Převzato z [2].

3.2.2 Notové linky odstraněny

Pokud jsou odstraněny notové linky, je nalezení izolovaných symbolů jednodušší a většinou stačí nalézt spojité segmenty. Některé symboly je ale potřeba dále zpracovat (rozdělení noty na hlavu a zbytek) a tak je potřeba v tomto kroku zjistit o jaký symbol se jedná. Hudební symboly lze zařadit do několika skupin podle tvaru a každá skupina symbolů je detekována a zpracována jinak. Následuje popis několika metod pro detekci a zpracování různých skupin symbolů.

- **Rossant F. [16]**
  Rossant dělí symboly na vertikální a ostatní. Vertikální symboly obsahují vertikální segment větší než $1.5d$. Jedná se o taktové čáry, noty (kromě celých), posuvky a některé mezery. Detekce těchto symbolů probíhá ve třech krocích:
  1. V každém sloupci je hledán vertikální segement větší než $1.5d$.
  2. Poté je detekován spojitý objekt obsahující nalezený vertikální segment.
  3. Nakonec je určen obalující obdélník nalezeného objektu. Tím jsou objekty izolovány a dojde i k rozdělení spojených not.

- **Capela A. a spol. [6]**
  Capela a spol. nejprve naleznou spojité segmenty. Tyto segmenty dále dělí do čtyř skupin (většinou podle tvaru):
  - **Složené noty**
    Pouze hledají segment o určité výšce propojující noty.
  - **Jednoduché noty**
    Definovali geometrické příznaky not na základě rozměrů. Výška je větší než $2d$ a šířka je od $0.5d$ do $3d$. Noty dále dělí na hlavu a zbytek.
  - **Posuvky, pomlky, time signatures**
    Detekovány pomocí kombinace horizontální a vertikální projekce. Počítají s tím, že rozměry těchto symbolů jsou v obou směrech podobné.

- **Klíče a vztahy mezi notami**
  Klíče mají velkou výšku a vztahy mezi notami mají velkou šířku. Na základě toho určily projekční profily, kdy klíč nabývá šířky $1d$ až $4d$ a výšky od $2d$ do $2 \cdot počet\ mezer \cdot d + počet\ linek \cdot n$ a vztahy mezi notami jsou podobně založeny na velké šířce.

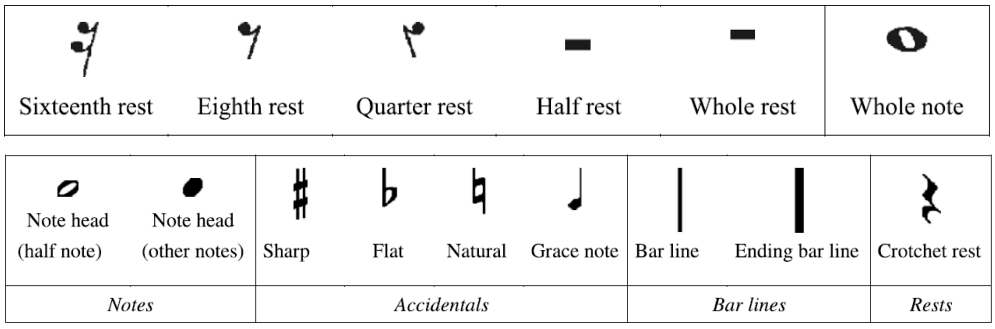
3.3 Klasifikace

Předposledním krokem OMR je klasifikace hudebních symbolů. Cílem klasifikace je rozdělit symboly do tříd určujících, o jaký symbol se jedná. Existuje mnoho přístupů a metod klasifikace, z nichž některé budou dále popsány.

- **Rossant F. [16]**
  Rossant zvolila jeden z asi nejjednodušších způsobů klasifikace objektů a to template matching. Vytvořila sadu vzorů pro všechny symboly (Obrázek 3.6). Vzory, se kterými se budou symboly porovnávat jsou vybrány na základě typu symbolu. Rossant dělí objekty na vertikální a ostatní. Ostatní symboly jsou porovnávány s jednou sadou vzorů. Vertikální objekty jsou rozděleny do čtyř skupin:

  1. Noty
  2. Taktové čáry
  3. Posuvky
  4. Pomlky

  Pro každou skupinu je vypočítána sada pěti kritérií na základě geometrie obalujícího obdélníku, vertikálního segmentu a pozice vztažené k notovým linkám. Pokud symbol splňuje tři z těchto kritérií, je zařažen do jedné ze skupin, na jejímž základě jsou vybrány vzory pro template matching.

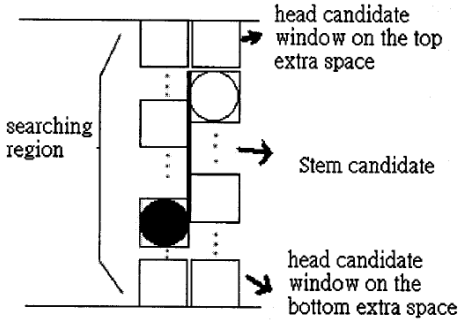


Obrázek 3.6: Rossant: vzory pro template matching. Nahoře vzory pro ostatní symboly, dole vzory pro vertikální symboly.
Převzato z [16].

- **Fujinaga I. [12]**
  Fujinaga se rozhodl pro klasifikaci pomocí k-nejbližších sousedů (k-NN), jelikož tento algoritmus umožňuje učit se nové třídy a s rostoucím množstvím dat se zlepšuje úspěšnost klasifikace. Pro každý symbol je nejprve vypočten vektor příznaků na základě měřitelných vlastností (šířka, výška, plocha, plocha obalujícího obdélníku, poměr stran,...). Dále je

vypočtena vzdálenost mezi novým symbolem a předchozími klasifikovanými symboly (na základě příznakových vektorů). Symbolu je přiřazena nejlépe odpovídající třída.

- **Hsin-Hua Chen a spol. [13]**
  Hsin-Hua Chen a spol. zvolili použití neuronových sítí, konkrétně HRCNN (two-layer Hyper Rectangular Composite Neural Network). Více o zvolené neuronové síti se lze dočíst v jejich práci. V té popisují použití pro notové hlavy. Nejprve si vytvoří příznakový vektor na základě rozměrů hlavy získaných horizontální a vertikální projekcí. Poté jsou natrénovány dvě neuronové sítě (pro plné a prázdné hlavy). Klasifikace probíhá tak, že testují kandidáty notových hlav na obou stranách notové nohy (Obrázek 3.7). Vypočítají projekční histogramy kandidátů a jejich podobnost s modely. Na základě této podobnosti je určeno, o jakou hlavu se jedná.



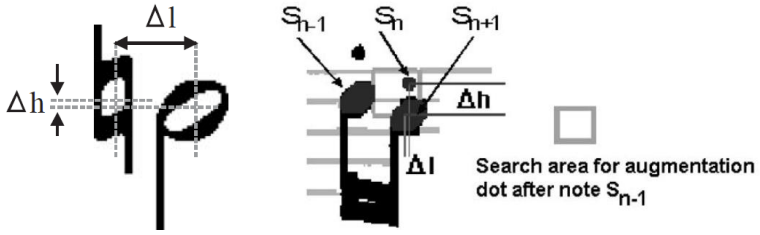
Obrázek 3.7: Hsin-Hua Chen a spol.: hledání kandidátů notových hlav.
Převzato z [13].

3.4 Post-processing

Post-processing je posledním krokem OMR a jeho cílem je detekovat chyby a nesrovnalosti vzniklé předchozími kroky. V tomto kroku se berou v úvahu hudební pravidla a na jejich základě je kontrolována správnost výsledku.

Rossant a Bloch [3] dělí pravidla na grafická a syntaktická. Grafická pravidla určují umístění hudebních symbolů v notové osnově, většinou pouze po sobě jdoucích, či k sobě vztažených. Definují umístění noty a její posuvky, noty a teček a různých doplňujících informací. Některá grafická pravidla zobrazuje Obrázek 3.8.

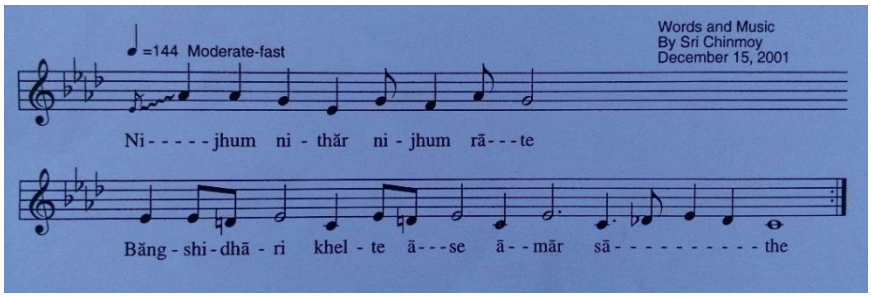
Syntaktická pravidla určují nějakou globální informaci platící pro celý notový zápis. Některá jsou striktní, jiná flexibilní a pouze zlepšují čtení notového zápisu. Hlavním syntaktickým pravidlem je počet dob v taktu. Další pravidla se týkají posuvek a předznamenání, kdy jsou počítány pravděpodobnosti po sobě jdoucích posuvek a popřípadě provedena korekce.



Obrázek 3.8: Grafická pravidla. Vlevo pozice noty a posuvky, vpravo pozice noty a tečky.
Převzato z [3].

4 Použité metody pro zpracování obrazu a klasifikaci

V této kapitole budou popsány metody pro práci s obrazem a klasifikaci používané v OMR. Bude popsáno předzpracování vstupního obrazu (převod z barevného prostoru), což je nezbytné pro další strojové zpracování (detekce hran, segmentace, ...). Obrázek 4.1 bude referenčním pro demonstraci některých metod. Důležitou částí je matematická morfologie, která v OMR nachází čím dál větší uplatnění. Bude ukázána metoda template matching, užívaná pro klasifikaci objektů, ale i robustnější přístupy k tomuto problému, jako je Support vector machines.



Obrázek 4.1: Zdrojový obrázek pro demonstrování výsledku některých metod zpracování obrazu.

4.1 Převod z barevného prostoru

Každý pixel barevného obrazu je reprezentován třemi hodnotami (jedna hodnota pro každou barevnou složku R, G a B). Tato reprezentace je vhodná pro lidské oko, ale pro strojové zpracování je příliš složitá. Už jen při pohledu na velikost takového obrazu, pokud intenzita každé barevné složky může nabývat hodnot z intervalu $< 0, 255 >$, je jeden pixelu reprezentován 24-mi bity. To je zbytečně mnoho, a proto existují metody, které zjednodušují reprezentaci obrazu a zároveň zachovávají veškeré podstatné informace v něm obsažené.

4.1.1 Převod do stupňů šedi

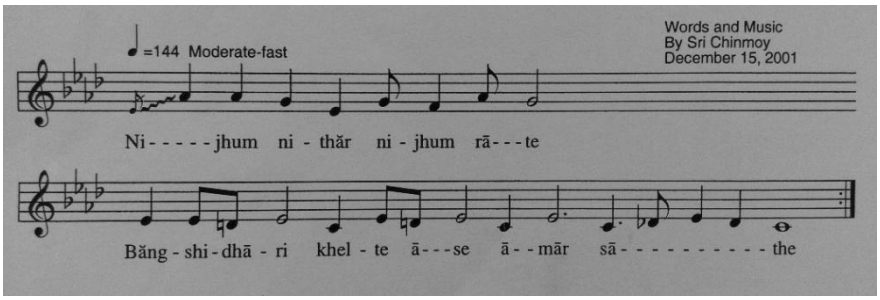
Převod do stupňů šedi, neboli grayscale, zjednodušuje reprezentaci barevného obrazu tím, že převádí hodnoty intenzity všech barevných složek každého pixelu na jednu hodnotu intenzity. Tím vznikne šedotónový obraz, jehož každý pixel je reprezentován jen jednou hodnotou, nejčastěji z intervalu $< 0, 255 >$ (Obrázek 4.2). Zároveň se neztrácí žádná podstatná informace, jelikož strojové zpracování obrazu většinou nepracuje s barvami, ale pouze s intenzitou pixelů.

Existuje více způsobů konverze barevného obrazu do stupňů šedi. Nejčastěji se ale používají následující dvě rovnice [7]:

$$\mathcal{G} = 0.299R + 0.587G + 0.114B \tag{1}$$

$$\mathcal{G} = 0.2126R + 0.7152G + 0.0722B \tag{2}$$

Obě rovnice jsou navrženy na základě lidského vnímání. Rovnice (1) je často využívána v počítačovém vidění a je implementována v mnoha programech, jako je MATLAB, GIMP, ... Rovnice (2) odpovídá více lidskému vnímání a je užívána v televizích s vysokým rozlišením [7].



Obrázek 4.2: Šedotónový obraz.

4.2 Detekce hran

Detekce hran je důležitou částí předzpracování obrazu. Hranové detektory hledají změny intenzity v obraze, jelikož hrany jsou pixely, kde se intenzita prudce mění [20].

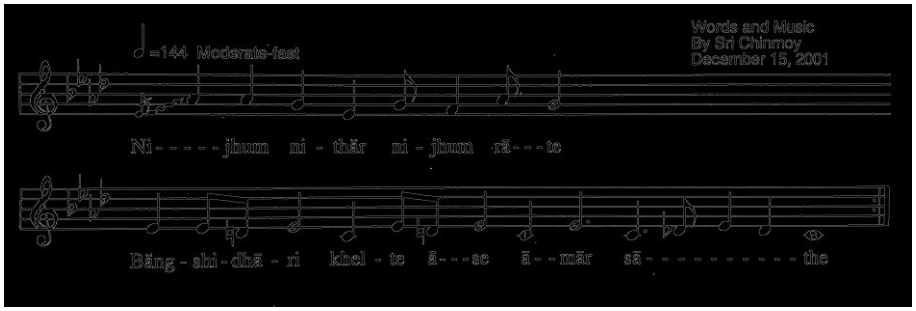
Výhodou detekce hran je další zjednodušení obrazu s tím, že jsou zachovány informace vedoucí k porozumění obsahu obrazu [20]. Hrany se často používají k detekci objektů v obraze.

4.2.1 Cannyho hranový detektor

Canny navrhl tři kritéria optimálního hranového detektoru [20]:

- **Detekce**: Důležité hrany nesmí být přehlédnuty a nesmí se objevit falešné detekce.
- **Lokalizace**: Vzdálenost detekované hrany a skutečné hrany by měla být minimální.
- **Jedna odezva**: Na jednu hranu by měla být jedna odezva. Pokud je více odezev na jednu hranu, jedna z nich je správná a ostatní jsou považovány za falešné.

Obraz je nejprve rozmazán Gaussovým filtrem a poté je vypočítána velikost a směr gradientu pro každý pixel rozmazaného obrazu. Směr gradientu je použit pro ztenčení hran potlačením pixelů, jejichž intenzita není větší než intenzita sousedních pixelů na obou stranách podél směru gradientu. To je nazýváno *nonmaximum suppression*. Dále jsou sledovány významné kontury. Sledování začíná na pixelu, jehož intenzita přesahuje vyšší práh a pokračuje dokud intenzita pixelu neklesne pod nižší práh [18]. Obrázek 4.3 zobrazuje výsledek této metody. Podrobnější informace uvadí Jain [14], Sonka [20], či Shapiro [18].



Obrázek 4.3: Hrany nalezené Cannyho hranovým detektorem.

4.3 Segmentace

Segmentace slouží k rozdělení obrazu do oblastí odpovídajících oblastem v zachycené scéně, nebo k rozlišení objektů zájmu [17].

4.3.1 Prahování

Prahování je nejjednodušší proces segmentace. Spočívá v nalezení prahu, podle nějž je obraz rozdělen na objekty a pozadí. Tím vznikne binární obraz.

Binární obraz f je mapování podmnožiny $\mathcal{D}_f \in Z^n$, zvané definiční obor obrazu f, do množiny $\{0, 1\}$:

$$f: \mathcal{D}_f \subset Z^n \rightarrow \{0, 1\}. \tag{3}$$

Tedy pro každý pixel p z definičního oboru obrazu, $f(p)$ odpovídá buď 0, nebo 1 [19].

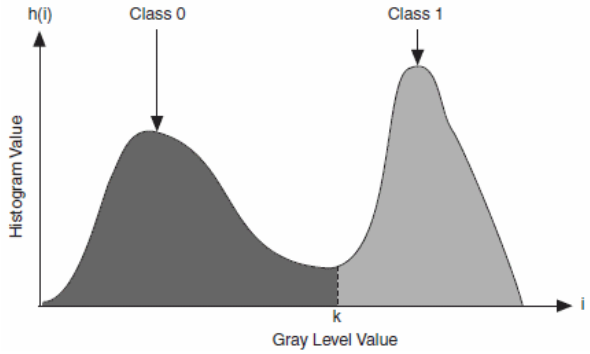
Prahování probíhá následujícím způsobem:

$$g(i,j) = 1 \; pro \; f(i,j) \geq T, \tag{4}$$

$$= 0 \text{ pro } f(i,j) < T,$$

kde T je práh, $g(i,j) = 1$ pro objekty a $g(i,j) = 0$ pro pozadí [20].

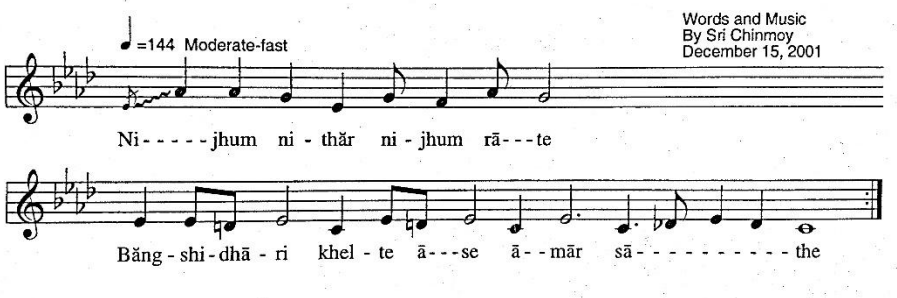
Kritickou částí je nalezení prahu, na němž záleží úspěšnost prahování. Práh je většinou zvolen na základě histogramu (Obrázek 4.4). Existují dva základní přístupy k nalezení prahu: globální prahování a adaptivní prahování.



Obrázek 4.4: Analýza histogramu.
Převzato z: http://zone.ni.com/reference/en-XX/help/372916L-01/nivisionconcepts/thresholding/

Globální prahování spočívá v nalezení jednoho prahu pro celý obraz, což málokdy vede k dobrému výsledku, jelikož v běžných obrazech dochází ke změnám intenzity v různých částech, způsobených například nerovnoměrným osvětlením.

Častěji se používá adaptivní prahování, kde se práh určuje na základě lokálních charakteristik obrazu. Jednou možností je rozdělení obrazu na části a určení prahu v každé části nezávisle. Pokud v nějaké části nemůže být práh určen, je interpolován z prahů v okolních částech. Každá část je poté prahována podle svého lokálního prahu [20]. Výsledek adaptivního prahování zobrazuje Obrázek 4.5.



Obrázek 4.5: Binární obrázek jako výsledek adaptivního prahování.

4.3.2 Houghova transformace

Houhgova transformace byla původně vynalezena pro sledování trajektorií subatomárních částic. Dnes se používá, mimo jiné, pro hledání transformovaných instancí předdefinovaných objektů v obraze. Nejčastěji se používá pro hledání přímek [9] (Obrázek 4.7).

Přímka je definována dvěma body $A = (x_1, y_1)$ a $B = (x_2, y_2)$. Všechny přímky procházející bodem A jsou popsány rovnicí:

$$y_1 = kx_1 + q, \tag{5}$$

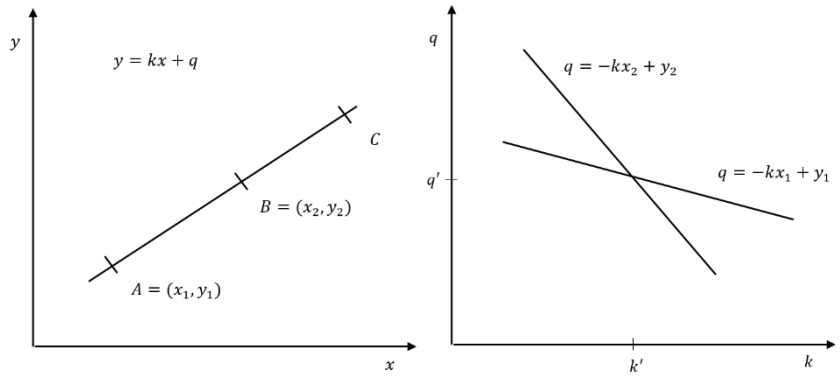
pro nějaké hodnoty k a q. To znamená, že tato rovnice může být interpretována jako rovnice v prostoru parametrů k a q. Všechny přímky procházející bodem A jsou poté popsány rovnicí:

$$q = -x_1 k + y_1. \tag{6}$$

Přímky procházející bodem B jsou posány podobně rovnicí:

$$q = -x_2 k + y_2. \tag{7}$$

Tyto přímky mají v prostoru parametrů k, q jediný průsečík reprezentující přímku v originálním obraze procházející body A a B (Obrázek 4.6).



Obrázek 4.6: Houghova transformace. Vlevo obrazový prostor, vpravo k, q parametrický prostor.

Počet přímek, procházejících bodem je nekonečný, což je ovšem pro praktické účely nepoužitelné. Proto je tento počet omezen, čímž je omezena i velikost parametrického prostoru, který může být reprezentován maticí buněk. Tato matice se nazývá akumulátor.

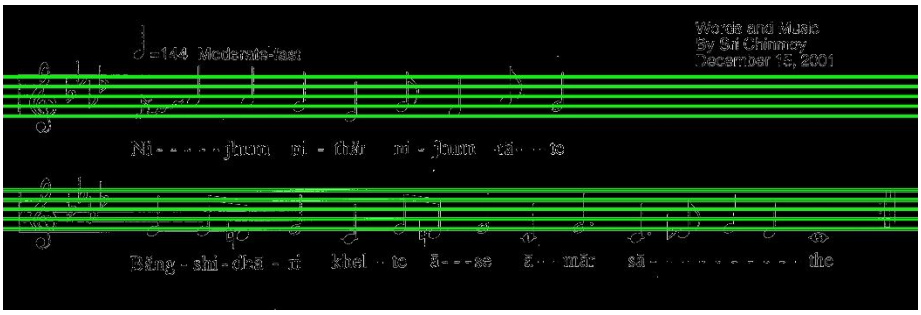
Každá přímka v originálním obraze je tedy reprezentována jedním bodem v akumulátoru. Hlavní myšlenkou je poté prozkoumat všechny body, které mohou tvořit přímku (většinou výstup hranového detektoru), tzn. transformovat všechny přímky, které mohou procházet těmito pixely do

odpovídajících bodů v akumulátoru. Přímky v původním obraze jsou poté reprezentovány jako lokální maxima v akulmulátoru.

Problémem při reprezentaci přímky rovnicí (5) je detekce vertikálních přímek, kdy $k \to \infty$. Proto je při reálném užití Houghovy transformace přímka reprezentována rovnicí:

$$s = x\cos\theta + y\sin\theta \tag{8}$$

a princip zůstává stejný [20].



Obrázek 4.7: Přímky nalezené Houghovou transformací.

4.4 Matematická morfologie

Matematická morfologie je založena na algebře nelineárních operátorů, pracujících s tvarem objektu a nahrazuje lineární algebraický systém konvoluce [20].

Je to nejrozsáhlejší třída operací pro zpracování binárních obrazů. Obsahuje erozi a dilataci a jejich kombinace a modifikace. Operace mohou být jednoduše popsány jako přidávání, či odstraňování pixelů z binárního obrazu na základě jistých pravidel, která závisí na okolních pixelech [17].

Každá operace je prováděna pro každý pixel obrazu jeho snímáním pomocí množiny známého tvaru, zvané strukturní element [19].

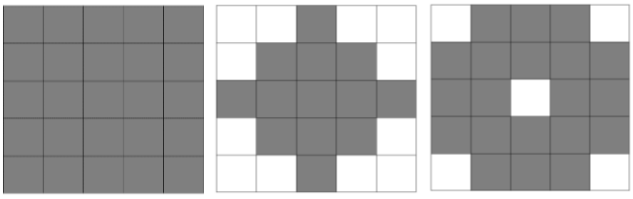
Matematická morfologie pracuje v mnoha oblastech zpracování obrazu (předzpracování, segmentace pomocí tvaru objektu, kvantifikace objektů) lépe a rychleji než standardní přístupy. Je často užívána v aplikacích, kde je problémem tvar objektů a rychlost, např. analýza obrázků z mikroskopu, kontrola v průmyslu, rozpoznávání textu, analýza dokumentů,... [20]

4.4.1 Strukturní element

Strukturní element je malá množina užitá k prozkoumání studovaného obrazu. Podgrafem n-rozměrného obrazu je $n+1$-rozměrná množina, proto $n+1$-rozměrný strukturní element může být použit ke zkoumání morfologie n-rozměrného obrazu. Avšak, aby nedocházelo k míchání rozměrových jednotek obrazu a jednotek intenzity odstínů šedi, používá se n-rozměrný strukturní element. Tyto strukturní elementy jsou označovány jako ploché, jelikož jsou 2-rozměrné (Obrázek 4.8).

Morfologické operátory vyžadují určení středu každého strukturního elementu. Tento střed umožňuje umístění strukturního elementu na určitý bod (pixel) v obraze: strukturní element v bodě x znamená, že jeho střed se shoduje s bodem x.

Tvar a rozměry strukturního elementu musí být přizpůsobeny geometrickým proporcím objektů v obraze, které mají být zpracovány. Například lineární strukturní element je užit při extrakci lineárních objektů. [19]



Obrázek 4.8: Příklady strukturních elementů.

4.4.2 Eroze

Eroze množiny X strukturním elementem B je značena $\varepsilon_B(X)$ a je definována polohami bodů x takových, že B je v x, tj. střed B se shoduje s x a B je obsaženo v X [19]:

$$\varepsilon_B(X) = \{x \mid B_x \subseteq X\}. \tag{9}$$

Implementace eroze může být zjednodušena tak, že množina X erodovaná strukturním elementem B může být vyjádřena jako průnik všech překladů X vektorem $-b \in B$ [20]:

$$\varepsilon_B(X) = \bigcap_{b \in B} X_{-b}. \tag{10}$$

Tato definice může být rozšířena na binární a šedotónové obrázky: eroze obrázku f strukturním elementem B je značena $\varepsilon_B(f)$ a je definována jako minimum z překladů f vektory $-b \in B$ [19]:

$$\varepsilon_B(f) = \bigwedge_{b \in B} f_{-b}. \tag{11}$$

Výsledná hodnota eroze v daném pixelu x je poté minimální hodnota obrazu v oblasti definované strukturním elementem se středem v x [19]:

$$[\varepsilon_B(f)](x) = \min_{b \in B} f(x + b). \tag{12}$$

Eroze se využívá ke zjednoduššení struktury objektu. Objekty, nebo jejich části s šířkou rovnou jedné zmizí. To může způsobit dekompozici složitého objektu na několik jednodušších [20]. Obrázek 4.9 vyobrazuje výsledek eroze.

Obrázek 4.9: Eroze. Vlevo původní obrázek, vpravo erodovaný obrázek.

4.4.3 Dilatace

Dilatace je duální operátor k erozi.

Dilatace množiny X strukturním elementem B je značena $\delta_B(X)$ a je definována polohami bodů x takových, že B je v x, tj. střed B se shoduje s x a B je obsaženo v X [19]:

$$\delta_B(X) = \{x \mid B_x \cap X \neq \emptyset\}. \tag{13}$$

Rovnice může být přepsána jako sjednocení množiny překladů X vektorem $-b \in B$ [19]:

$$\delta_B(X) = \bigcup_{b \in B} X_{-b}. \tag{14}$$

Tato definice může být rozšířena na binární a šedotónové obrázky: dilatace obrázku f strukturním elementem B je značena $\delta_B(f)$ a je definována jako maximum z překladů f vektory $-b \in B$ [19]:

$$\delta_B(f) = \bigvee_{b \in B} f_{-b}. \tag{15}$$

Výsledná hodnota dilatace v daném pixelu x je poté maximální hodnota obrazu v oblasti definované strukturním elementem se středem v x [19]:

$$[\delta_B(f)](x) = \max_{b \in B} f(x + b). \tag{16}$$

Dilatace je využívána k zaplnění malých děr a úzkých zálivů v obraze. Zvětšuje rozměry objektů [20]. Výsledek dilatace zobrazuje Obrázek 4.10.

Obrázek 4.10: Dilatace. Vlevo původní obrázek, vpravo dilatovaný obrázek.

4.4.4 Morfologické otevření a uzavření

Eroze a dilatace nejsou inverzní transformace. Pokud byl obraz erodován, následnou dilatací není získán původní obraz. Místo toho je výsledek zjednodušená a méně detailní verze originálního obrazu [20].

- **Otevření**
  Myšlenkou morfologického otevření je dilatace erodovaného obrazu za účelem co nejlepší rekonstrukce původního obrazu (Obrázek 4.11uprostřed).

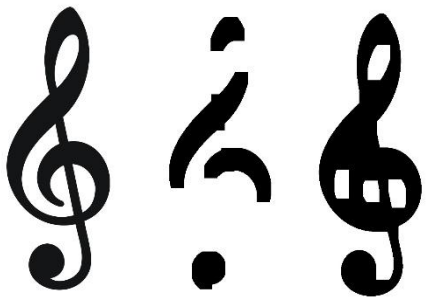
  Otevření γ obrazu f strukturním elementem B je značeno $\gamma_B(f)$ a je definováno jako eroze f elementem B následovaná dilatací f stejným elementem [19]:

$$\gamma_B(f) = \delta_B[\varepsilon_B(f)]. \tag{17}$$

- **Uzavření**
  Myšlenkou morfologického uzavření je vytvořit operátor směřující k obnovení původního tvaru struktur v dilatovaném obraze. Toho je dosaženo pomocí eroze dilatovaného obrazu (Obrázek 4.11vpravo).

  Uzavření obrazu f strukturním elementem B je značeno $\phi_B(f)$ a je definováno jako dilatace f elementem B následovaná erozí f stejným elementem [19]:

$$\phi_B(f) = \varepsilon_B[\delta_B(f)]. \tag{18}$$



Obrázek 4.11: Otevření a uzavření. Vlevo původní obrázek, uprostřed po aplikaci otevření, vpravo po aplikaci uzavření. Strukturní elementem byl čtverec o velikosti $40\ x\ 40$ px.

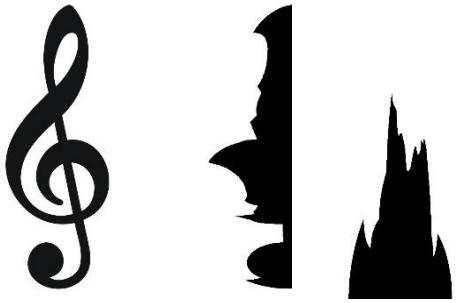
4.5 Projekce binárních obrazů

Projekce jsou kompaktní reprezentace binárních obrazů. Obsahují důležité informace, užívané pro popis regionů v obraze. Projekce ovšem nejsou unikátní, více obrazů může mít stejnou projekci.

Projekce binárního obrazu na úsečku může být získána rozdělením obrazu na části (řádky, sloupce,...) a hledáním počtu černých pixelů v těchto částech [14].

Horizontální a vertikální projekce $p_h(i)$ a $p_v(j)$ jsou definovány jako [20] (Obrázek 4.12):

$$p_h(i) = \sum_j f(i,j), \qquad p_v(j) = \sum_i f(i,j). \tag{19), (20}$$



Obrázek 4.12: Projekce. Vlevo původní obrázek, uprostřed horizontální projekce, vpravo vertikální projekce.

4.6 Run-length encoding

Run-length encoding (RLE) je kompaktní reprezentace binárního obrazu pomocí čísel popisujících délky spojitých úseků černých a bílých pixelů v daném směru. Existují dva způsoby zápisu výsledného kódu [14]:

- Zápis pomocí dvojic:
  $(pozice\ počátečního\ černého\ pixelu, počet\ černých\ pixelů\ v\ úseku)$

- Zápis pouze pomocí délek úseků. Začíná se černými pixely a střídají se délky úseků černých pixelů a bílých pixelů.

4.7 Klasifikace

Klasifikace slouží k rozdělení dat do tříd. Nejčastěji se využívá k rozpoznávání objektů. V této práci je využita k rozpoznávání hudebních symbolů, kdy každý symbol je přiřazen k jedné z předem známých tříd.

Existuje mnoho metod a přístupů ke klasifikaci. Jedním z nich je metoda Support vector machines (použita v této práci), která je popsána dále. Dalším hodně využívaným přístupem je užití neuronových sítí. Ty jsou popsány například Sonkou [20], Shapiroem [18], nebo Forsythem [11]. Je zde zahrnuta i metoda template matching, která se v OMR často využívá pro klasifikaci hudebních symbolů.

4.7.1 Template matching

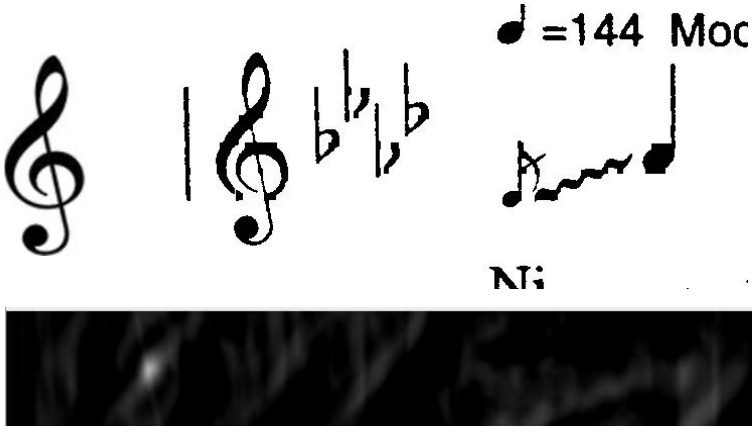
Template matching je metoda detekce předem známých objektů v obraze. K tomu je potřeba obraz, ve kterém mají být objekty detekovány a vzor (template), neboli (většinou) malý obrázek s hledaným objektem. Vzor často vzniká vyříznutím malé oblasti z nějakého obrazu.

Metoda poté postupně posouvá vzor po všech možných pozicích v obraze a počítá jeho shodu s oblastí odpovídající aktuální pozici vzoru.

Výpočet může probíhat různými způsoby. Většinou se jedná o porovnávání intenzit pixelů vzoru a odpovídající části obrazu, ale mohou být využity i různé obrazové příznaky a deskriptory [20]. Většinou se používá kros-korelace vyjádřena rovnicí [17]:

$$R(i,j) = \frac{\sum_{x=0}^{s}\sum_{y=0}^{s} B_{x+i,y+j} \cdot T_{x,y}}{\sqrt{\sum_{x=0}^{s}\sum_{y=0}^{s} B_{x+i,y+j}^2 \cdot \sum_{x=0}^{s}\sum_{y=0}^{s} T_{x,y}^2}}, \tag{21}$$

kde B je intenzita pixelu v obraze, T je intenzita pixelu ve vzoru, s je velikost vzoru a i, j jsou souřadnice aktuálně zkoumaného pixelu. V tomto případě ve výsledku (Obrázek 4.13 dole) hledáme lokální maximum označující nejlepší shodu (levý horní roh templatu).



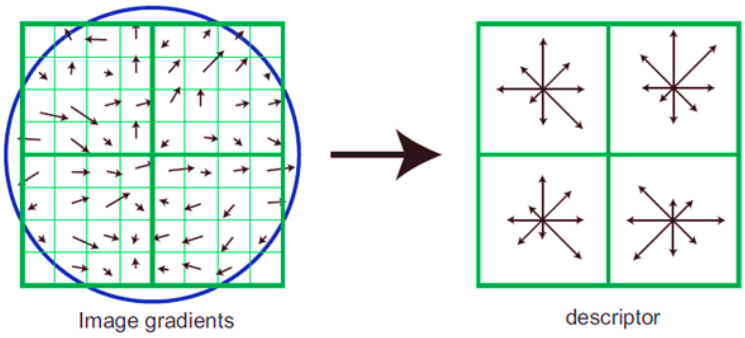
Obrázek 4.13: Template matching. Vlevo nahoře je použitý vzor. Vpravo nahoře je prohledávaný obraz. Dole je výsledek, nejsvětlejší bod určuje nejlepší shodu.

4.7.2 Histogram orientovaných gradientů

Histogram orientovaných gradientů má za úkol získat příznaky v obraze vyhodnocením normalizovaných lokálních histogramů orientací gradientů v husté mřížce, které jsou použity při klasifikaci. Hlavní myšlenka spočívá v tom, že lokální vzhled a tvar objektu může být charakterizován distribucí lokální intenzity gradientů či orientací hran a to i bez přesné znalosti odpovídajících gradientů nebo pozic hran.

V praxi je to provedeno rozdělením obrazu do malých prostorových regionů (buněk). Pro každou buňku (přes všechny její pixely) je akumulován lokální 1D histogram směrů gradientu nebo orientací hran. Reprezentace obrazu je dána kombinací těchto buněk.

Pro větší invarianci vůči změnám osvětlení či stínování, atd., je užitečné normalizovat i lokální odezvy, před jejich použitím. To může být provedeno akumulací „energií“ lokálních histogramů přes větší regiony obrazu (bloky) a použitím výsledků k normalizaci všech buněk v bloku. Tyto bloky se nazývají deskriptory Histogramu orientovaných gradientů (Obrázek 4.14). [8]



Obrázek 4.14: Histogram orientovaných gradientů.
Převzato z: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

4.7.3 Support vector machines

Metoda Support vector machines (dále SVM) pracuje ve vektorovém prostoru a jejím cílem je oddělit dvě třídy datových bodů tzv. nadrovinou (hyperplane). Následující popis SVM pro lineárně oddělitelná data vychází z popisu vytvořeném Forsythem [11].

Je dána množina N bodů $\vec{x}_i$ patřících do dvou tříd $(1, -1)$. Body jsou označeny značkami tříd y_i, datová sada může být poté zapsána následovně:

$$\left\{\left(\vec{x}_1, y_1\right), \ldots, \left(\vec{x}_N, y_N\right)\right\}.$$

Je třeba najít pravidlo, které určí značku třídy pro jakýkoliv bod $\vec{x}$ (klasifikátor).

Cílem je určit parametry $\vec{w}$ a b (popisují oddělující nadrovinu) pro každý vzorový bod tak, že platí:

$$y_i\left(\vec{w} \cdot \vec{x}_i + b\right) > 0. \tag{22}$$

Existuje množina omezení na výběr parametrů $\vec{w}$ a b. Tato omezení určují, že všechny vzorové body s negativní značkou y_i by měly být na jedné straně nadroviny a všechny vzorové body s pozitivní značkou y_i by měly být na druhé straně nadroviny. Každá tato nadrovina musí oddělovat konvexní obálku jedné množiny vzorových bodů od konvexní obálky druhé množiny vzorových bodů. Nejlepší

nadrovina je ta, která je co nejvíce vzdálena od obou zmíněných obálek. Toho je docíleno tak, že jsou spojeny nejbližší body z obou obálek a středem této spojnice je vedena nadrovina.

Nyní lze zvolit měřítko parametrů $\vec{w}$ a b (změna měřítka nemá vliv na vztah (22)). To znamená, že lze zvolit $\vec{w}$ a b takové, že pro každý datový bod platí:

$$y_i\left(\vec{w}\cdot\vec{x}_i+b\right)\geq 1, \tag{23}$$

přičemž rovnost je dosažena pro alespoň jeden bod na každé straně nadroviny. Platí-li pro $\vec{x}_k$ rovnost a $y_k=1$ a platí-li pro $\vec{x}_l$ rovnost a $y_l=-1$, znamená to, že $\vec{x}_k$ je na jedné straně nadroviny a $\vec{x}_l$ na straně druhé. Navíc vzdálenost těchto bodů je od nadroviny minimální. Takových bodů může být více, proto $\vec{w}\cdot\left(\vec{x}_1-\vec{x}_2\right)=2$, tudíž:

$$dist\left(\vec{x}_k, nadrovina\right)+dist\left(\vec{x}_l, nadrovina\right) \tag{24}$$

$$=\left(\frac{\vec{w}}{\left|\vec{w}\right|}\cdot\vec{x}_k+\frac{b}{\left|\vec{w}\right|}\right)-\left(\frac{\vec{w}}{\left|\vec{w}\right|}\cdot\vec{x}_l+\frac{b}{\left|\vec{w}\right|}\right)$$

$$=\frac{\vec{w}}{\left|\vec{w}\right|}\cdot\left(\vec{x}_1-\vec{x}_2\right)=\frac{2}{\left|\vec{w}\right|}.$$

Z toho vyplývá, že maximalizovat vzdálenost je stejné jako minimalizovat vztah $(1/2)\vec{w}\cdot\vec{w}$, čímž dostáváme omezený problém minimalizace: minimalizovat vztah $(1/2)\vec{w}\cdot\vec{w}$ vzhledem k podmínce (23).Tento problém lze vyřešit Langrangeovou funkcí:

$$\frac{1}{2}\vec{w}\cdot\vec{w}-\sum_1^N\propto_i\left(y_i\left(\vec{w}\cdot\vec{x}_1+b\right)-1\right). \tag{25}$$

Tato funkce musí být minimalizována vzhledem k $\vec{w}$ a b a maximalizována vzhledem k $\propto_i$. Tím získáme dva požadavky:

$$\sum_1^N\propto_i y_i=0, \qquad \vec{w}=\sum_1^N\propto_i y_i\vec{x}_i. \tag{26), (27}$$

Díky druhému požadavku je tento klasifikátor znám jako Support vector machine. Obecně je nadmnožina určena malým počtem vzorových bodů a pozice ostatních vzorových bodů je nepodstatná. To znamená, že většinou je parametr $\propto_i$ nulový a datové body odpovídající nenulovým $\propto_i$ (ty určují nadrovinu) jsou označeny jako podpůrné vektory (support vectors).

Po dosazení požadavků (26) a (27) do funkce (25), je získán dvojí optimalizační problém, popisující řešení SVM:

$$\begin{aligned} &maximalizovat && \sum_i^N\propto_i-\frac{1}{2}\sum_{i,j=1}^N\propto_i\left(y_iy_j\vec{x}_i\cdot\vec{x}_j\right)\propto_j\\ &vzhledem\ k && \propto_i\geq 0\\ &a && \sum_1^N\propto_i y_i=0. \end{aligned}$$

5 Aplikace rozpoznávání a přehrávání not z fotografie

V této kapitole bude prezentována výsledná aplikace. Nejprve bude popsán návrh aplikace s ohledem na cíle a požadavky a budou popsány jednotlivé metody použité pro rozpoznání a přehraní not v obraze. Budou zmíněny problémy, které se vyskytly během vývoje a bude popsáno jejich řešení. V sekci implementace bude popsána struktura programu.

5.1 Cíle a požadavky

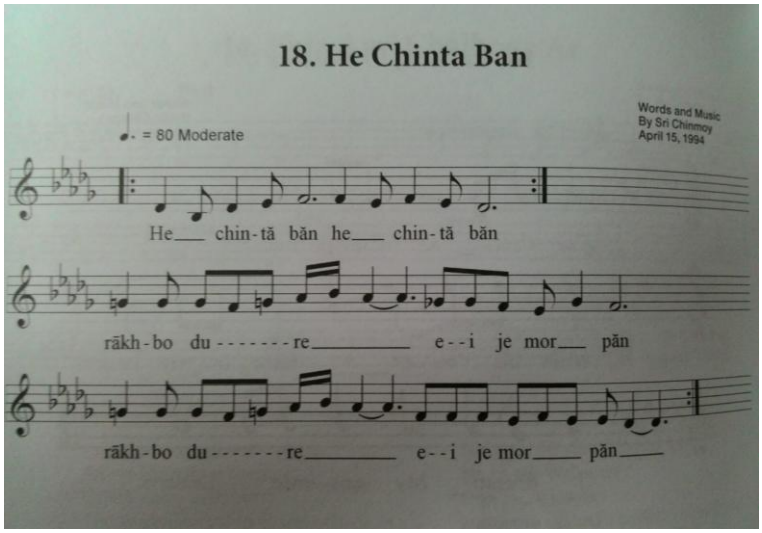
Jedním z cílů aplikace je běh v reálném čase. Mobilní telefony zatím nedosahují výkonu stolních počítačů či notebooků, tudíž lze čekat, že výsledek bude k dispozici v řádu jednotek minut.

Uživatelské rozhraní aplikace by mělo být jednoduché a intuitivní a mělo by obsahovat pouze nezbytné ovládací prvky.

Výsledná aplikace by tedy měla umožnit uživateli načíst obrázek (z paměti telefonu či pomocí fotoaparátu), tento obrázek zpracovat, přehrát a uložit výslednou hudbu.

5.1.1 Vstupní data

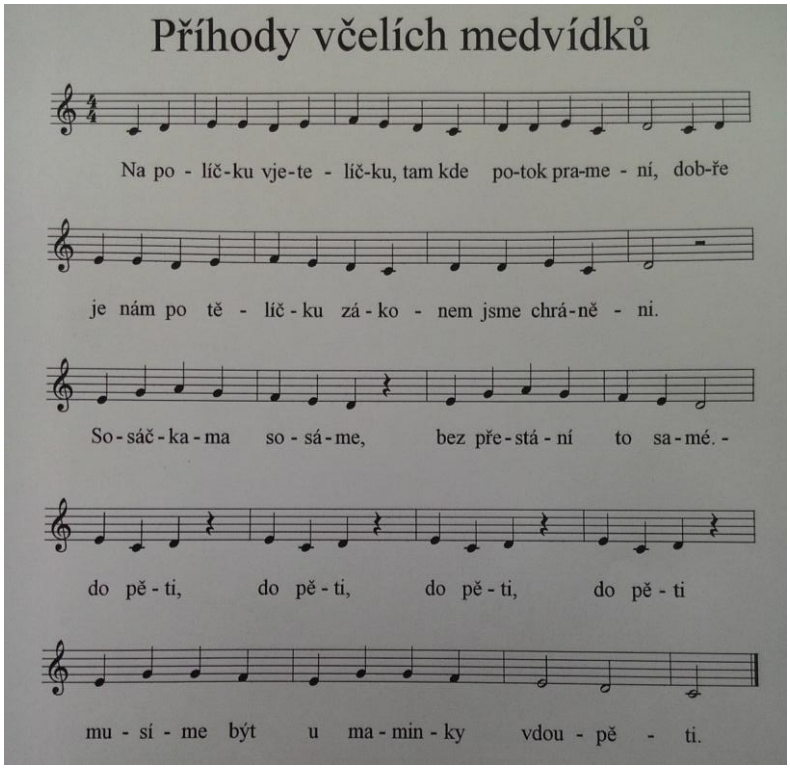
Jak bylo zmíněno, vstupními daty jsou obrázky pořízené fotoaparátem mobilního telefonu, či načtené z paměti. Bohužel většina mobilních telefonů nemá příliš kvalitní fotoaparát a obrázky takto pořízené často podléhají různým geometrickým deformacím. Přidáli se k tomu navíc nekvalitně vytištěný notový zápis, pak je zpracování takového obrazu značně obtížné. Dalšími problémy mohou být zdeformovaný papír, který je focen, špatné osvětlení či rozmazání. Obrázek 5.1 zobrazuje příklad takovéto fotografie. V dalších kapitolách bude popsáno řešení těchto problémů takové, aby bylo dosaženo co nejlepších výsledků. Většinou ovšem takový obrázek nemůže být správně zpracován.



Obrázek 5.1: Příklad nekvalitního vstupního obrázku.

Na základě znalosti výše zmíněných problémů můžeme definovat ideální vstupní obraz. Ten by měl obsahovat pouze notový zápis, notové linky by měly být vodorovné a co nejméně zdeformované. Obraz by měl být ostrý a osvětlení by mělo být rovnoměrné. Obrázek 5.2 ukazuje příklad ideálního vstupního obrazu.

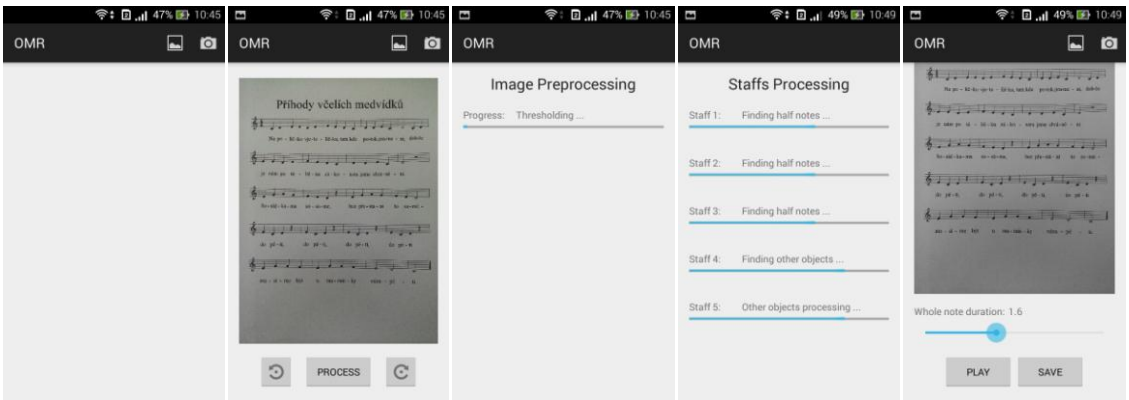
Je zřejmé, že takto popsaný obraz se nepovede vždy pořídit a proto musí aplikace umožnit zpracovat jakýkoliv obraz s tím, že pokud dojde k chybě během zpracování, musí o tom být uživatel informován.



Obrázek 5.2: Příklad ideálního vstupního obrazu.

5.2 Uživatelské rozhraní

Při návrhu uživatelského rozhraní byl kladen důraz na jednoduchost a informativní charakter. Uživatel má tedy k dispozici pouze nezbytné ovládací prvky a během zpracovávání je informován o jeho průběhu. Obrázek 5.3 zobrazuje obrazovky výsledné aplikace tak, jak za sebou následují.



Obrázek 5.3: Grafické rozhraní výsledné aplikace.

Na úvodní obrazovce má uživatel možnost načíst obrázek. Může zvolit obrázek z paměti telefonu, nebo použít fotoaparát.

Po načtení obrázku je potřeba obrázek otočit do přirozené polohy tak, aby byly linky vodorovné. K tomu slouží dvě tlačítka pro otočení o 90° doleva a doprava. Ke spuštení zpracování obrazu slouží tlačítko „PROCESS“.

Po spuštění zpracování je uživatel informován o průběhu pomocí výpisů a indikátorů průběhu. Nejprve je zobrazen průběh předzpracování obrazu a poté průběh zpracování jednotlivých osnov.

Nakonec má uživatel možnost zvolit délku celé noty v rozmezí od 0,1s do 4,0s a přehrát či uložit výsledný zvuk.

5.3 Návrh aplikace

Aplikace byla navržena s ohledem na rozdělení problému OMR na podproblémy prezentované v kapitole 3. Tyto podproblémy jsou:

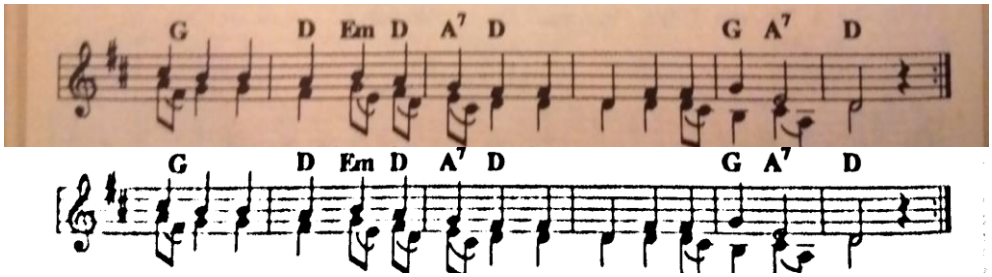
- Předzpracování dat
- Segmentace
- Klasifikace
- Post-processing

Dále budou probrány jednotlivé podproblémy a bude popsán způsob, jakým byly vyřešeny. Budou zde popsány problémy, které se během implementace vyskytly a způsob, jakým byly vyřešeny.

5.3.1 Předzpracování dat

Prvním problémem při zpracování obrázků z různých zařízení je různé rozlišení. Proto je potřeba nejprve změnit velikost obrázku na nějakou uniformní hodnotu. Byla zvolena šířka obrázku 2000 pixelů, což je dostatečná velikost při ohledu na poměr *rychlost/úspěšnost zpracování*. Výška obrázku je dopočítana na základě poměru stran.

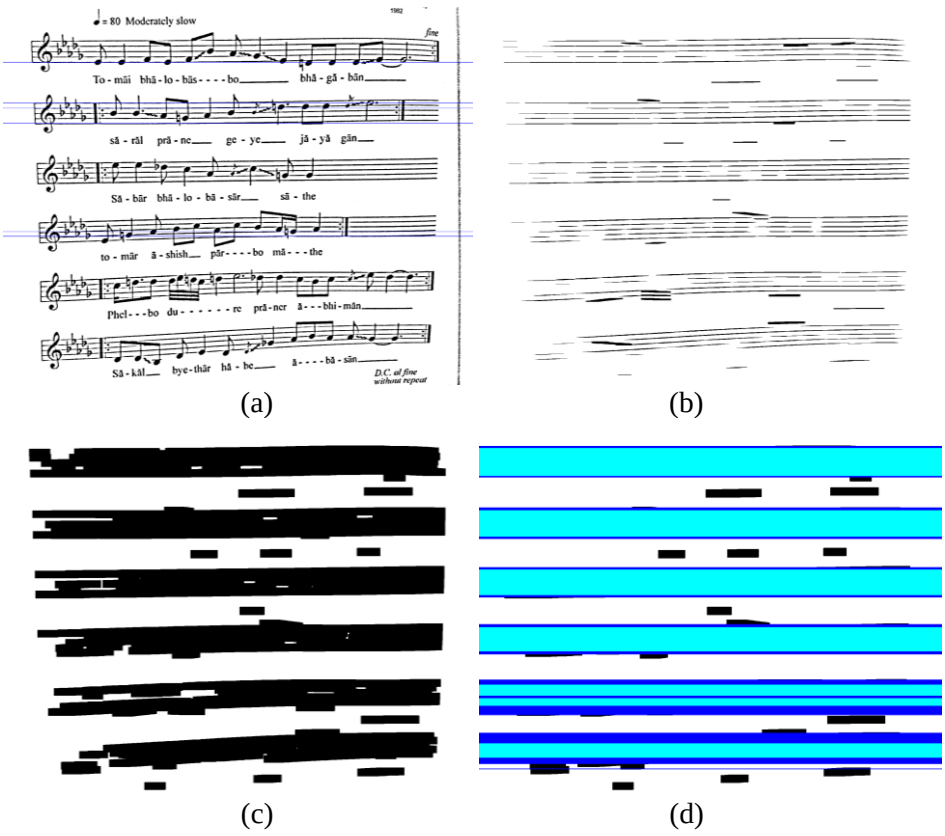
Po změně velikosti následuje binarizace obrazu. Bylo použito adaptivní prahování s Gaussovským jádrem o velikosti 101 pixelů, což se po četných testech ukázalo jako nejvhodnější řešení. V případech, kdy je obrázek rozmazaný bohužel binarizace neproběhne bez problému a dojde k rozbití notových linek a některých symbolů (Obrázek 5.4). Proto je potřeba mít na vstupu obrázek zaostřený.



Obrázek 5.4: Binarizace rozmazaného obrázku.

Další částí je nalezení notových osnov. Nejprve byla použita horizontální projekce pro nalezení notových linek a poté byly vyhledány pětice těchto linek označujících notovou osnovu. Tento postup se ovšem ukázal jako nevhodný, jelikož funguje pouze, pokud jsou notové linky vodorovné a nejsou zdeformované. Proto je vhodný zejména pro kvalitně naskenované obrázky. Jak bylo zmíněno

v sekci 5.1.1, obrázky pořízené fotoaparátem mobilního telefonu nejsou ideální a notové linky jsou často deformované, což znemožní jejich detekci při použití pouze horizontální projekce. Obrázek 5.5 (a) zobrazuje výsledek použití horizontální projekce pro detekci notových linek. Jak lze vidět, byly detekovány pouze některé linky. Proto bylo potřeba vymyslet jinou metodu pro detekci linek a osnov. Byla tedy použita matematická morfologie. Nejprve je aplikováno morfologické otevření s jádrem o velikosti $1\ x\ 50$ pixelů (50 pixelů dlouhá vodorovná čára), díky kterému zůstanou v obraze pouze horizontální čáry (Obrázek 5.5 (b)). Tyto čáry jsou spojeny do jednolitých celků pomocí dilatace s jádrem $30\ x\ 30$ pixelů (Obrázek 5.5 (c)). Tyto jednolité celky označují jednotlivé osnovy v obrázku. Osnovy jsou poté detekovány horizontální projekcí (Obrázek 5.5 (d)) a jsou vyříznuty. Následuje paralelní zpracování izolovaných notových osnov.



Obrázek 5.5: Detekce notových osnov. Obrázek (a) zobrazuje výsledek horizontální projekce. Obrázek (b) zobrazuje výsledek morfologického otevření s jádrem $1\ x\ 50$ pixelů. Obrázek (c) zobrazuje výsledek dilatace s jádrem $30\ x\ 30$ pixelů. Na obrázku (d) je výsledek horizontální projekce na dilatovaném obrázku.

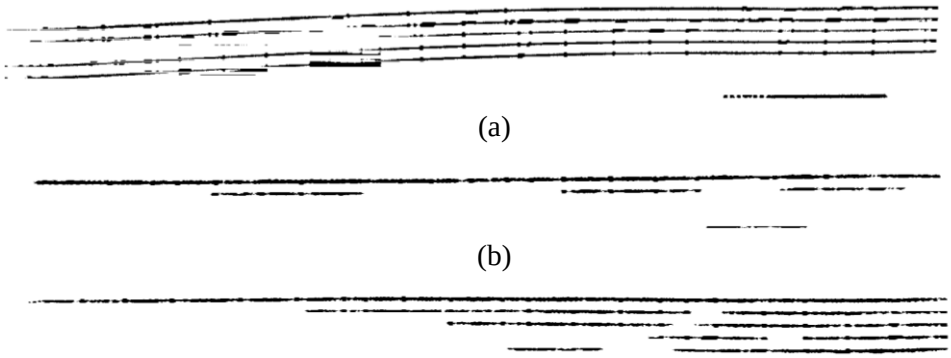
Notové osnovy je potřeba nejprve zarovnat, tj. natočit je tak, aby byly notové linky co nejvíce vodorovné. K tomu je nejprve použita Houghova transformace, která nalezne čáry v obraze. Jedná se o pravděpodobnostní Houghovu transformaci, která vrací seznam souřadnic krajních bodů nalezených čar. Jednotlivé nalezené čáry jsou procházeny a je počítána jejich délka. Nejdelší čára je uložena a podle ní je osnova natočena. Nejprve je spočítán úhel rotace na základě rozdílu y-ových souřadnic krajních bodů a pomocí funkce arkus tangens je vypočítán výsledný úhel, o který je osnova natočena. Natočení probíhá vynásobením matice s osnovou rotační maticí.

Následuje kritická část OMR, kterou je detekce a odstranění notových linek. Jak bylo řečeno, notové linky v obrázcích pořízených fotoaparátem nelze detekovat pomocí jednoduchých metod, jako je horizontální projekce. Proto je potřeba zvolit robustnější metody. Při detekci notových osnov dobře posloužily morfologické operace, proto byly použity i při detekci notových linek. Nejprve je aplikována dilatace s jádrem $5\ x\ 5$ pixelů pro usnadnění dalšího kroku. Tímto krokem je aplikace

morfologického otevření s jádrem $1\ x\ 150$ pixelů (150 pixelů dlouhá vodorovná čára). Tím dojde k detekci horizontálních čar v obraze (Obrázek 5.7 (b)). Toto ještě nejsou finální notové linky a je potřeba vyfiltrovat černé pixely, které skutečně leží na notových linkách. Toho je docíleno součtem matice s notovou osnovou a matice s detekovanými čarami. Tím je získána finální maska obsahující pouze pixely ležící na notových linkách. Tuto masku zobrazuje Obrázek 5.7 (c).

Bohužel ani tato metoda se neobejde bez chyb a může dojít k nesprávnému vytvoření masky. Příklady chybně vytvořených masek zobrazuje Obrázek 5.6. K tomuto jevu dochází hlavně ve dvou případech:

- Notové linky jsou příliš zdeformované. K tomu dochází zejména kvůli deformaci papíru, který je focen.
- Notové linky nejsou spojité a obsahují díry. K tomu může dojít při binarizaci obrazu (jak bylo zmíněno výše), nebo pokud je osnova špatně vytištěna a poté vyfocena.



(a)

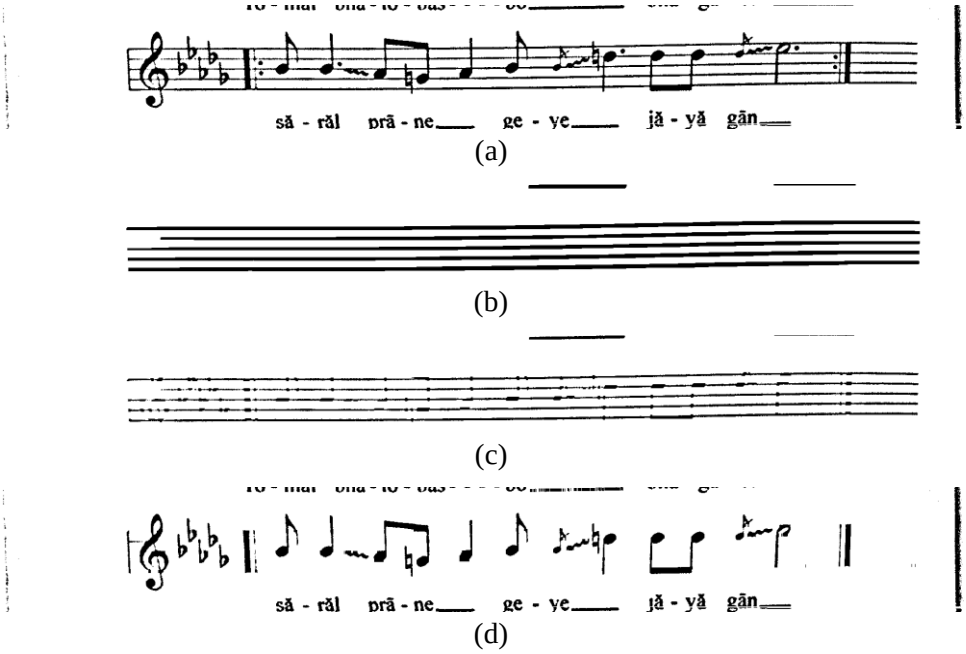
(b)

(c)

Obrázek 5.6: Chybně vytvořené masky notových linek.

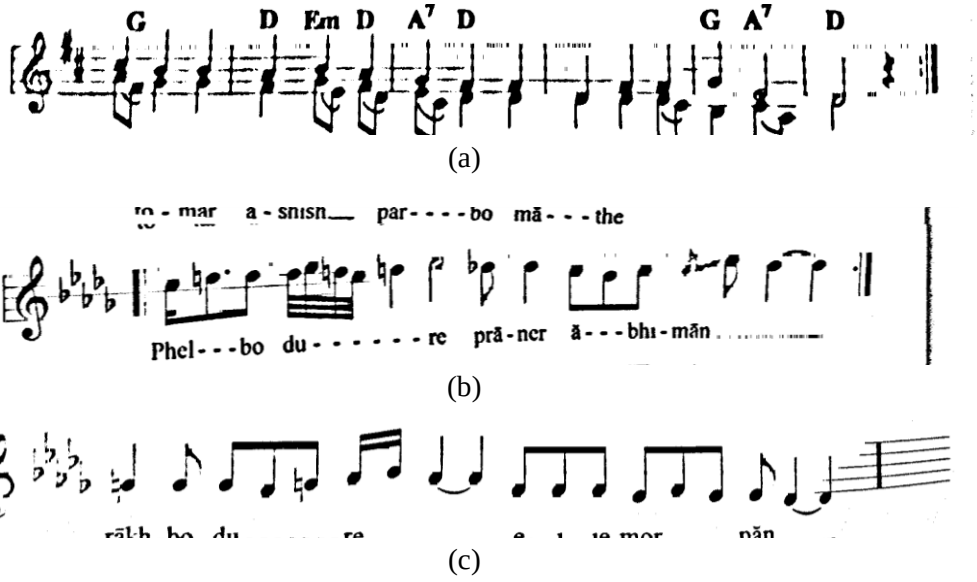
Před odstraněním detekovaných linek je potřeba získat z masky dva důležité údaje popisující notovou osnovu. Těmito údaji jsou průměrná tloušťka notových linek a průměrná velikost mezery mezi linkami. V každém sloupci masky jsou hledány černé a bílé segmenty pixelů (značící linky a mezery). Průměrná tloušťka linek je snadno vypočítána ze všech detekovaných černých segmentů. U detekce průměrné mezery mezi linkami je potřeba nejprve zpracovat nalezené mezery v každém sloupci, jelikož se v osnově nachází i segmenty bílých pixelů, které neznačí mezery mezi linkami. Tyto mezery je potřeba odstranit. Nejprve je odstraněna první detekovaná mezera, jelikož to je většinou mezera od horního okraje obrázku k první notové lince. Dále jsou odstraněny všechny mezery, které jsou menší než 5 pixelů. Pokud zbyly přesně čtyři mezery, jsou jejich velikosti zahrnuty do finálního výpočtu. V případě, kdy maska není vytvořena správně a v žádném sloupci není pět linek, zpracování dané osnovy vrátí chybový kód a uživatel je informován o tom, že obrázek nelze zpracovat.

Následuje odstranění deteketovaných notových linek. Jsou procházeny černé pixely v masce a na odpovídající pozici v původním obrázku s notovou osnovou je zkoumána velikost vertikálního segmentu černých pixelů. Pokud je tato velikost menší než $průměrná\ tloušťka\ linky + 2$, je tento segment odstraněn. Výsledek odstranění notových linek zobrazuje Obrázek 5.7 (d).



Obrázek 5.7: Detekce a odstranění notových linek. Obrázek (a) zobrazuje původní notovou osnovu. Na obrázku (b) je výsledek morfologického otevření s jádrem $1 \; x \; 150$ pixelů. Na obrázku (c) je maska obsahující pouze pixely, které leží na notových linkách. Obrázek (d) zobrazuje notovou osnovu s odstraněnými notovými linkami.

V případě chybně vytvořené masky samozřejmě dojde i k chybnému odstranění notových linek. Příklady chybně odstraněných linek zobrazuje Obrázek 5.8. Pokud chyb není příliš mnoho (Obrázek 5.8 (b), (c)), další metody zpracovávající osnovu jsou chopny si s tímto ve většině případů poradit. V opačném případě (Obrázek 5.8 (a)) nelze očekávat slušný výsledek.

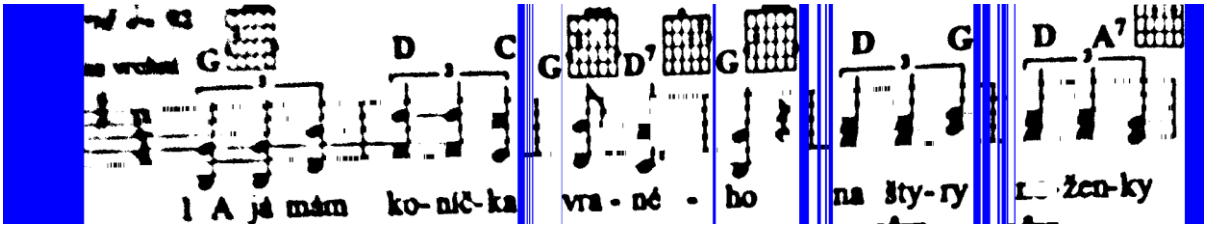


Obrázek 5.8: Chybně odstraněné notové linky.

5.3.2 Segmentace

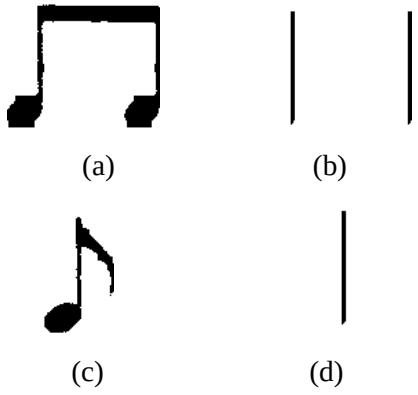
Segementace je dalším krokem OMR. Jejím cílem je nalézt jednotlivé hudební symboly v osnově s odstraněnými notovými linkami. Osnova na vstupu již neobsahuje některé symboly, které byly detekovány pomocí metody template matching. Jedná se o tyto symboly: tečky, celé a půlové noty, klíče a posuvky. Způsob, jakým byly tyto symboly zpracovány bude popsán v sekci Klasifikace 5.3.3. V této sekci bude uveden způsob detekce zbylých symbolů, jejich rozdělení na vertikální a nevertikální a jejich příprava ke klasifikaci.

Prvním krokem je detekce objektů. K tomu slouží vertikální projekce, která hledá prázdné sloupce v obraze, oddělující od sebe jednotlivé objekty. Ty jsou z osnovy vyříznuty a dále zpracovány. V případě chybně odstraněných linek (viz předchozí kapitola) nelze objekty oddělit a v dalších krocích nejsou správně zpracovány. V některých zápisech jsou navíc nad osnovou různé značky, které znemožňují oddělení symbolů. Příklad takové osnovy zobrazuje Obrázek 5.9.



Obrázek 5.9: Chybná detekce objektů.

Následuje rozdělení objektů na vertikální a nevertikální. Vertikální objekty jsou noty a taktové čáry. Nevertikální jsou všechny ostatní objekty. Vertikálními jsou objekty označeny proto, že obsahují vertikální segement černých pixelů větší, než je určitý práh. V tomto případě byl práh určen jako $3{,}5 \cdot (průměrná\ velikost\ mezery\ mezi\ linkami)$. Tento segment je v objektu detekován pomocí morfologických operací. Nejprve je aplikována dilatace s jádrem $2\ x\ 2$ pixelů k odstranění malých děr v objektu, které mohli vzniknout při binarizaci, či při odstranění linek. Následuje morfologické otevření s jádrem $3{,}5 \cdot (průměrná\ mezera\ mezi\ linkami)\ x\ 1$. Po této operaci zůstane v obraze, v případě vertikálního objektu, minimálně jedna vertikální čára (Obrázek 5.10 (b), (d)). V případě nevertikálního objektu je výsledný obraz prázdný. Pozice vertikálních segmentů jsou poté detekovány pomocí vertikální projekce a jsou uloženy a využity při dalším zpracování.



Obrázek 5.10: Detekce vertikálních objektů. Na obrázku (a) a (c) je zobrazen původní objekt. Na obrázku (b) a (d) je poté výsledek morfologického otevření.

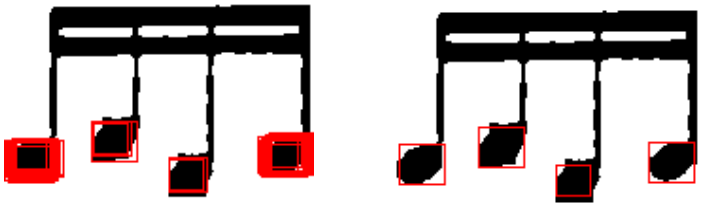
Oba druhy objektů je potřeba dále zpracovat a připravit ke klasifikaci. Zpracování nevertikálních symbolů spočívá v nalezení největšího spojitého segmentu černých pixelů, který vyhovuje zadaným podmínkám. Nejprve jsou pomocí detekce kontur nalezeny všechny spojité segmenty černých pixelů. Dále jsou zkoumány vlastnosti ohraničujících obdélníků nalezených segmentů. Pokud tyto vlastnosti nevyhovují jedné z následujících podmínek, je segment zahozen. Podmínky jsou:

- Plocha ohraničujícího obdélníku je menší než $(mezera\ mezi\ linkami)^2/2$.
- Y-ová souřadnice středu ohraničujícího obdélníku je menší než $(počet\ řádků\ objektu)/3$.
- Y-ová souřadnice středu ohraničujícího obdélníku je větší než $(počet\ řádků\ objektu\ /\ 3)\cdot 2$.

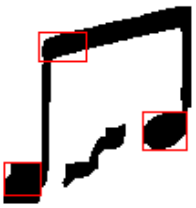
Pokud nezbyl žádný segment, objekt není klasifikován. Jedná se většinou o nepořádek, který v osnově zůstal po odstranění linek. Pokud zbyl právě jeden segment, je z obrazu vyříznut. V případě, že zbylo více segmentů je nalezen ohraničující obdélník, který je všechny obsahuje a tento obdélník je vyříznut.

Pokud je objekt rozbitý, může dojít k tomu, že žádný z jeho segmentů nesplňuje výše uvedené podmínky a objekt není klasifikován. Občas se to stává u osminových pomlk

Zpracování vertikálních objektů je poněkud složitější. Nejprve je potřeba určit, zda se jedná o notu, nebo taktovou čáru. Proto jsou v objektu detekovány notové hlavy. Detekce probíhá pomocí metody template matching (4.7.1). Pro detekci hlav jsou využity tři vzory (templates) (Obrázek 5.17). Každý vzor je užit v několika velikostech vztažených k průměrné velikosti mezery mezi linkami. Parametry pro detekci notových hlav zobrazuje Tabulka 1. Ve výsledné matici, kterou vrací metoda template matching (viz Obrázek 4.13) je detekována pozice maximální hodnoty. Pokud je maximální hodnota menší než práh, vzor v obraze nebyl nalezen. V opačném případě je pozice maximální hodnoty uložena a je smazána z výsledné matice. Takto jsou v cyklu nalezena veškerá maxima nacházející se v dané výsledné matici. Tato maxima označují pozice detekovaných notových hlav a je potřeba je dále zpracovat. Téměř vždy je jedna hlava detekována několika vzory v různých velikostech. Proto je pro každou hlavu vybrán pouze výsledek s největší pravděpodobností (Obrázek 5.11). Dále jsou odstraněny výsledky jejichž pozice v x-ovém směru se liší od dříve detekovaných pozic vertikálních segementů. To jsou chybně detekované hlavy, vyskytující se převážně u složených not. Bohužel pokud pozice takto chybně detekované hlavy odpovídá pozici některého z vertikálních segementů (Obrázek 5.12), nelze ji označit za chybně detekovanou a je uložena mezi nalezené hlavy. Byly provedeny různé experimenty se vzory a parametry metody template matching, ale bohužel se nepodařilo chybnou detekci hlav zcela odstranit.



Obrázek 5.11: Mnohonásobná detekce hlav a výsledek její filtrace.



Obrázek 5.12: Chybně detekovaná hlava.

Následuje analýza nalezených notových hlav. Pokud nebyla nalezena žádná hlava a šířka objektu je menší než $(průměrná\ mezera\ mezi\ linkami)^2/3$, je objekt označen jako taktová čára a je uložen do seznamu objektů. V opačném případě je potřeba rozhodnout, zda se jedná o jednoduchou, či složenou notu. Pokud je počet hlav právě jedna, pak se jedná o jednoduchou notu. Pokud je hlav víc, nelze ihned říct, že se jedná o složenou notu, jelikož v případě akordů může jednochá nota obsahovat více notových hlav. Proto je potřeba prozkoumat pozici všech nalezených hlav. Pokud je x-ová pozice všech hlav téměř stejná, jedná se o jednoduchou notu. V opačném případě se jedná o notu složenou.

Zpracování jednoduchých not probíhá následujícím způsobem. Nejprve je nalezen největší spojitý segment černých pixelů pomocí detekce kontur a tento segment je vyříznut. Následuje odstranění detekovaných notových hlav, jelikož klasifikovat se bude pouze notová noha s praporem. Obrázek 5.13 zobrazuje zpracovanou jednoduchou notu.

Obrázek 5.13: Zpracování jednoduché noty.

Složené noty je potřeba nejdříve rozdělit a jednotlivé části jsou zpracovány stejně jako noty jednoduché. Rozdělení složené noty probíhá v cyklu, kdy je brána vždy první notová hlava ze seznamu detekovaných hlav a je vyříznuta část složené noty odpovídající šířce této hlavy (plus 10 pixelů na obě strany). Dále jsou ze seznamu všech detekovaných hlav odstraněny ty hlavy, které se vyskytují ve vyříznuté části. Tato část je poté zpracována stejným způsobem jako jednoduchá nota. Obrázek 5.14 zobrazuje složenou notu a části, na které byla rozdělena.

Obrázek 5.14: Rozdělení složené noty.

Všechny zpracované objekty postupují do klasifikace. Nejprve ale musí být upraveny, jelikož každý klasifikátor vyžaduje určitý formát vstupních dat. V tomto případě je potřeba obrázek objektu převést na čtverec o velikosti $32\ x\ 32$ pixelů. Nejprve jsou přidány okraje tak, aby z objektu vznikl čtverec. Poté je objekt zvětšen na $128\ x\ 128$ pixelů a je aplikováno Gaussovo rozmazání s jádrem o velikosti

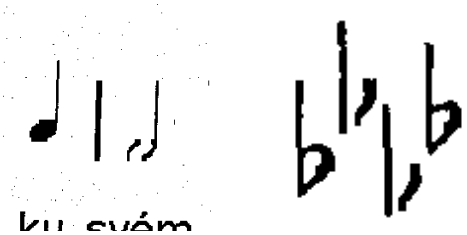
$7\ x\ 7$ pixelů a nulovou standardní odchylkou. Následně je objekt zmenšen na velikost $32\ x\ 32$ pixelů, čímž je připraven ke klasifikaci. Obrázek 5.15 zobrazuje příklady objektů připravených ke klasifiaci.

Obrázek 5.15: Příklady objektů připravených ke klasifikaci.

5.3.3 Klasifikace

Klasifikace byla rozdělena do dvou kroků. První krok probíhá ještě před segmentací a je při něm použita metoda template matching. V druhém kroku jsou klasifikovány objekty, nalezené při segementaci, pomocí metody SVM. Následuje popis obou kroků.

Template matching
Původně bylo zamýšleno zpracovat všechny objekty způsobem popsaným v sekci Segmentace 5.3.2 a poté je klasifikovat pomocí SVM. Bohužel se ihned ukázalo, že to není možné. Při pohledu na Obrázek 5.16 lze vidět, že některé objekty jsou často rozbity při odstranění linek. Navíc symboly v předznamenání jsou tak blízko sebe, že mezi nimi většinou není žádná mezera. Z toho důvodu tyto symboly není možné detekovat pomocí vertikální projekce a proto byla zvolena metoda template matching.



Obrázek 5.16: Symboly rozbité při odstranění linek.

V kapitole 3.3 bylo ukázáno, jak Rossant F. [16] používá template matching pro klasifikaci všech objektů. V jejím případě jde ovšem o obrázky naskenované, s jedním typem fontu a zpracování probíha na počítači. Za těchto podmínek je tuto metodu možné použít, jelikož není potřeba velké množství vzorů a zpracování proběhne rychle.

V případě této práce tuto metodu nelze použít při klasifikaci všech objektů. Je to dáno několika faktory, které byly zmíněny v předchozím textu. Jedná se o tyto faktory, které se navzájem ovlivňují:

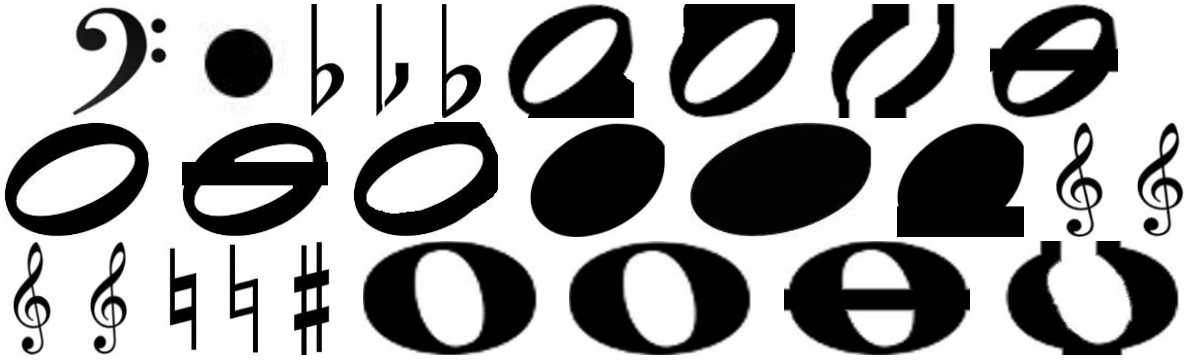
- Vstupní data
- Rozbité symboly a různé fonty
- Běh v reálném čase

V případě vstupních dat bylo řečeno, že se jedná o obrázky neideální, podléhající různým geometrickým deformacím a nepravidelnému osvětlení. Z toho důvodu se kvalitou nemohou srovnávat s obrázky naskenovanými a proto při zpracování dochází k chybám (špatná binarizace, špatně odstranění linek).

Jelikož dochází k chybám při zpracování obrazu, často dochází k rozbití některých symbolů (viz výše). Proto v metodě template matching nelze použít pouze jeden vzor a je potřeba vzorů několik. To stejné platí v případě různých fontů.

Při použití několika vzorů se zvyšuje doba zpracování. To se znatelně projevuje hlavně na starších mobilních zařízeních a poté nelze mluvit o běhu v reálném čase, což je jedním z hlavních požadavků na výslednou aplikaci.

Z těchto důvodů byla metoda template matching použita pouze při detekci klíče, celých a půlových not, teček a posuvek. Bylo vytvořeno velké množství vzorů a během implementace a testování byla snaha toto množství co nejvíce zredukovat. Finální počet vzorů je pořád velký, ale všechny jsou potřeba a další snížení jejich počtu by vedlo ke zhoršení úspěšnosti. Obrázek 5.17 zobrazuje všechny použité vzory.

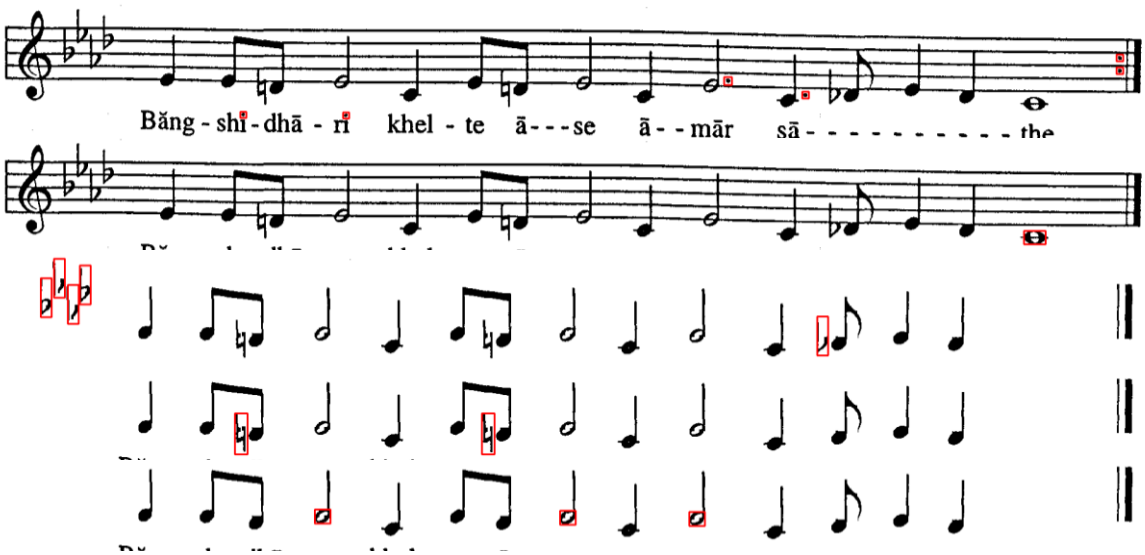


Obrázek 5.17: Vzory použité v metodě template matching.

Klasifikace metodou template matching probíhá následujícím způsobem. Nejprve jsou načteny vzory podle objektu, který má být detekován. Následně jsou nastaveny parametry, které ukazuje Tabulka 1. Další postup popisuje Algoritmus 1. Každý vzor je detekován v několika velikostech vztažených k průměrné velikosti mezery mezi linkami. Ve výsledku metody template matching (Obrázek 4.13) je poté detekována maximální hodnota. Pokud je tato hodnota větší než zadaný práh, její pozice je uložena do seznamu detekovaných objektů. To probíhá v cyklu, kdy jsou nalezeny všechny pozice v daném výsledku, jejichž hodnota je větší než práh. Jak bylo zmíněno již v sekci 5.3.2, kde byla tato metoda použita pro detekci notových hlav, téměř vždy dochází k mnohonásobné detekci jednoho objektu a je proto potřeba vzít pouze tu detekovanou pozici, která má největší pravděpodobnost.

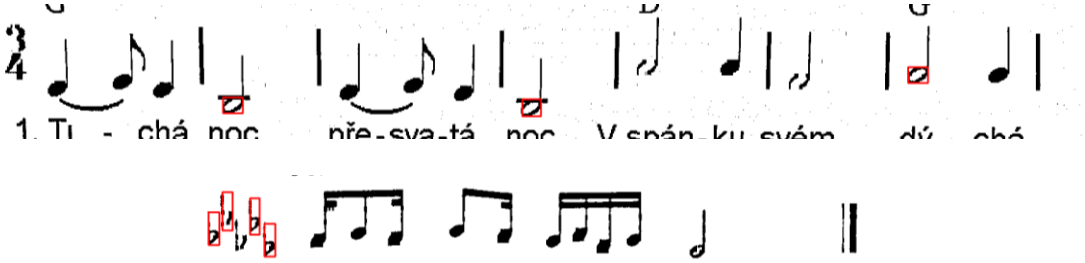
Všechny takto detekované objekty jsou uloženy do seznamu objektů v osnově. Je uložen jejich typ a pozice významného bodu, který slouží při určování výšky objektu. Tento bod je většinou střed obdélníku obalujícího daný objekt. U béček je y-ová souřadnice této pozice rovna y-ové souřadnici spodního okraje tohoto obdélníku. Dále jsou objekty z osnovy smazány (pixely obalujícího obdélníku v osnově jsou nastaveny na bílou barvu). Obrázek 5.18 zobrazuje příklady detekce objektů.

Lze vidět, že některé objekty (tečky a celé noty) jsou detekovány ještě před odstraněním linek. V případě teček je to proto, že po odstranění linek mohou v obraze zůstat různé malé segmenty černých pixelů, které by poté mohly být označeny jako tečka. Na stejném obrázku lze ovšem vidět, že i přesto k chybné detekci teček dochází a je potřeba tečky zpracovat. Zpracování teček bude popsáno v kapitole 5.3.4. Celé noty jsou detekovány před odstraněním linek z důvodu redukce počtu vzorů. Byly provedeny různé experimenty i s půlovými notami. Bohužel v tomto případě nedošlo k redukci počtu vzorů a proto jsou půlové noty detekovány až po odstranění čar.



Obrázek 5.18: Objekty detekované metodou template matching.

V průběhu implementace a testování této metody muselo být zavedeno několik kompromisů, aby bylo dosaženo optimálního poměru *doba zpracování/úspěšnost*. Obrázek 5.19 zobrazuje chyby při detekci objektů. Jak lze vidět, některé objekty nebyly detekovány (dvě půlové noty nahoře a jedno béčko dole). Stává se tak z důvodu již několikrát zmíněného a tím je nekvalitní vstupní obraz vedoucí k chybám při zpracování a rozbití objektů. K eliminaci tohoto jevu, jak je řečeno výše, bylo vytvořeno mnoho vzorů. Bohužel, aby byly pokryty všechny případy a byly detekovány veškeré objekty, muselo by být použitých vzorů daleko víc, čímž by ovšem opět nebyl splněn požadavek na běh aplikace v reálném čase. Již nyní je doba vykonávání této metody průměrně 45% celkové doby zpracování obrazu. Proto bylo nutné se smířit s tím, že tato metoda nebude stoprocentní a u nekvalitních obrázků bude docházet k nedetekování některých objektů, čímž je samozřejmě poté ovlivněn výsledný zvuk.



Obrázek 5.19: Ukázka nedetekovaných objektů metodou template matching.

Typ objektu	Velikost*			Práh	
	od	do	krok	první	druhý
Tečka	0,6	0,9	0,1	0,8	-
G klíč	6,0	12,0	1,0	0,5	-
F klíč	3,0	6,0	1,0	0,5	-
Béčko	3,6	4,3	0,2	0,75	-
Odrážka	3,0	5,2	0,2	0,7	-
Křížek	3,0	4,5	0,2	0,6	-
Celá nota	1,0	1,5	0,1	0,68	0,8**
Půlová nota	1,3	1,6	0,1	0,62	-
Plná hlava	1,1	1,5	0,1	0,67	-

Tabulka 1: Parametry pro detekci objektů metodou template matching.
*Výška vzoru je vypočítána jako $velikost \cdot$ průměrná $mezera\ mezi\ linkami$,
šířka je dopočítána na základě poměru stran.
**Tento práh je použit pro velikosti 1,0 a 1,1 z důvodu redukce počtu
falešných detekcí například v textu písně pod osnovou.

Algoritmus 1 Detekce objektů metodou template matching

For ($velikost = od;\ velikost < do;\ velikost \mathrel{+}= krok$) {
  For ($i = 0;\ i <$ počet vzorů$;\ i++$) {
    $vzor$ = pole vzorů na pozici i;
    $poměr\ =$ počet sloupců $vzoru$/počet řádků $vzoru$;
    $velikost\ vzoru =$
    $= (velikost \cdot mezera\ mezi\ linkami \cdot poměr)\ x\ (velikost \cdot mezera\ mezi\ linkami)$;
    změna velikosti vzoru na základě $velikost\ vzoru$;
    $výsledek\ =\ template\ matching$; //nalezení vzoru v obraze
    while (pravda) {
      $max =$ detekce maximální hodnoty ve $výsledek$;
      if $max < práh$
        break;
      ulož max do seznamu detekovaných objektů;
      smaž max z výsledku;
    }
  }
}

SVM

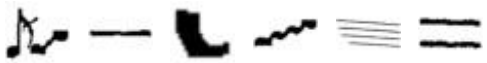
Metoda SVM je užita ke klasifikaci objektů, které jsou výstupem segmentace. Příklady těchto objektů zobrazuje Obrázek 5.15. Byla využita knihovna LIBSVM, která poskytuje jednoduché rozhraní jak pro samotnou klasifikaci, tak i pro natrénování vlastních modelů. K tomu je potřeba dataset obsahující obrázky všech objektů. Nebyl nalezen žádný volně dostupný dataset a proto byl vytvořen vlastní.

Tvorba obrázků pro dataset probíhala stejně jako zpracování objektů popsané v sekci 5.3.2, tzn. nafocené notové zápisy byly zpracovány a obrázky objektů byly uloženy a roztříděny. Bylo vytvořeno 44 tříd a ty byly rozděleny do tří skupin:

- Jednoduché noty – čtvrťové až čtyřiašedesátinové.
- Složené noty – osminové až čtyřiašedesátinové.
- Ostatní objekty – pomlky celé až čtyřiašedesátinové, časové značky: běžný čas, 2/4, 3/4, 4/4 a další objekty, které se mohou vyskytovat v osnově, ale nejsou pro tuto práci důležité byly označeny jako „nic“. Jejich příklad zobrazuje Obrázek 5.21.

Možných objektů v notovém zápisu je více, ovšem v této aplikaci jsou uvažovány pouze tyto základní. Byla snaha mít u jednotlivých tříd co nejvíce obrázků. Bohužel některé objekty se v notových zápisech vyskytují více, jiné méně. Proto je u některých tříd obrázků málo, což ovlivní následnou klasifikaci. Více bude popsáno v sekci Vyhodnocení 6.3. Příklady obrázků z datasetu zobrazuje Obrázek 5.20.

Obrázek 5.20: Ukázka obrázků z datasetu.



Obrázek 5.21: Ukázka objektů označených jako „nic“.

Po vytvoření datasetu byly natrénovny tři modely. Nejprve bylo potřeba vytvořit příznakový vektor pro každý obrázek. Pro výpočet příznaků byl zvolen histogram gradientů (Histogram of gradients – HOG). HOG je popsán v sekci 4.7.2. Poté je převeden na příznakový vektor, který má tvar: `číslo_třídy 0:hodnota0 1:hodnota1...` Tyto vektory jsou spojeny do jednoho souboru, ze kterého je poté natrénován model.

Natrénování modelu je v knihovně LIBSVM velmi snadné. Je pouze nutné zvolit určité parametry. Byl zvolen multitřídní klasifikátor (C-SVC) s jádrem RBF (radial basis function). Pro toto jádro je potřeba určit parametr gamma a dále je potřeba určit hodnotu cenové funkce (C). Tyto parametry jsou získány pomocí cross validace, během níž jsou testovány všechny kombinace zvolených parametrů a je získána ta, s níž dosahuje natrénovaný model nejlepší úspěšnosti. Pro cross validaci je v knihovně obsažen skript `grid.py`, který byl spuštěn s těmito parametry:

```
./grid.py -log2c -10,20,2 -log2g 10,-20,-2 -v 5
```

Výsledkem byly nejlepší hodnoty parametrů C a gamma, které byly využity při následeném trénování modelu. To proběhlo pomocí utility `svm-train`, která je opět zahrnuta v knihovně LIBSVM.

Pro klasifikaci je využita knihovna jlibsvm, což je rozhraní LIBSVM pro jazyk Java. Nejprve je opět vypočítán histogram gradientů, který je převeden na příznakový vektor. Ten je v jlibsvm polem struktur `svm_node`. Do každé této struktury je uložen index příznaku a jeho hodnota. Toto pole (vektor příznaků) je předáno funkci `svm_predict()` spolu s potřebným modelem. Tato funkce vrací číslo (label) třídy, značící typ objektu. Tento objekt je poté uložen do seznamu všech objektů v dané osnově.

5.3.4 Post-processing

Nyní jsou veškeré objekty nalezené v dané osnově uloženy v seznamu objektů. K tomu, aby z nich bylo možné vytvořit zvuk je potřeba je nejdříve zpracovat. V této sekci bude popsáno, jak je získána

výška jednotlivých symbolů a jak probíhá zpracování teček a posuvek. Dále bude zdůvodněno, proč je vynechána kontrola počtu dob v taktu, která se většinou v OMR aplikacích vyskytuje.

Určení výšky symbolů

Výška symbolů je nezbytná pro vytvoření zvuku. Je určována pro každý symbol zvlášť podle pozice jeho významného bodu, což je většinou střed obdélníku, který ho obaluje (u not obdélník obaluje pouze hlavu). Aby bylo možné určit výšku symbolu, je potřeba znát pozici notových linek v osnově. Z důvodu deformace notových linek není možné určit jejich pozici pro celou osnovu a poté určovat výšku symbolů. Je potřeba pro každý symbol určit pozici linek v jeho okolí. Nejprve jsou linky hledány v okolí o šířce 40 pixelů. Pokud nedojde ke správné detekci a určení výšky symbolů, je toto okolí rozšířeno na 100 pixelů a linky jsou detekovány znovu. Pokud ani v tomto případě nedojde ke správné detekci, jsou použity pozice linek detekovaných u posledního z předchozích symbolů, u kterého byly správně detekovány. Problém nastává, pokud u žádného z předchozích symbolů nebyly linky správně detekovány. V tom případě nemůže být určena výška symbolu.

Detekce linek probíhá následujícím způsobem. Nejprve jsou detekovány linky v celé osnově stejným způsobem, jako při odstranění linek (5.3.1), tj. pomocí morfologických operací. Nejprve je aplikována dilatace s jádrem $5\ x\ 5$ pixelů a poté je užito morfologické otevření s jádrem $1\ x\ 150$ pixelů. To je provedeno jednou pro celou osnovu. Obrázek 5.22 zobrazuje takto detekované linky. Následuje vyříznutí části tohoto obrázku, která odpovídá zadanému okolí zpracovávaného objektu. V této části jsou linky detekovány pomocí horizontální projekce. Linky je potřeba dále zpracovat. Nelze určovat výšku symbolu, pokud není k dispozici právě pět linek. Pokud je linek šest, jedna linka je odstraněna. Je to buď první nebo poslední linka, podle velikosti mezery k další nejbližší lince. Pokud je tato mezera větší než průměrná mezera, je linka odstraněna. Pokud v tuto chvíli není linek pět, je vrácen chybový kód a funkce je zavolána znovu s širším okolím. V opačném případě je přidáno dalších šest linek – tři pomocné linky nad osnovou a tři pomocné linky pod osnovou. Jsou podporovány pouze výšky v tomto rozmezí. Obrázek 5.25 zobrazuje všechny podporované výšky. Výška symbolu je poté určena podle y-ové pozice významného bodu mezi těmito linkami.

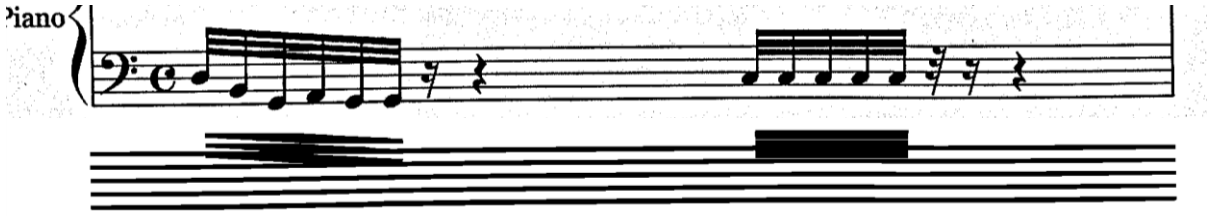
Obrázek 5.22: Linky detekované v osnově při určování výšky symbolů.

Při určování výšky symbolů dochází ke dvěma problémům:

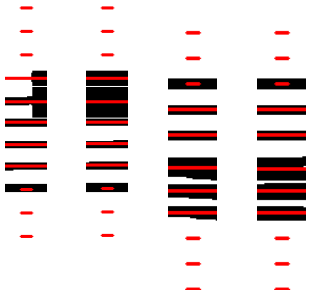
- Výška nemůže být určena.
- Výška je určena chybně.

Výška někdy nemůže být určena z důvodu zmíněného výše. Tzn. nebylo detekováno pět linek v okolí širokém 40 pixelů, nebylo detekováno pět linek ani při rozšíření okolí na 100 pixelů a u žádného z předchozích symbolů nebylo detekováno pět linek. Symboly s nedetekovanou výškou jsou při vytváření zvuku vynechány, čímž je ovlivněn celý výsledek.

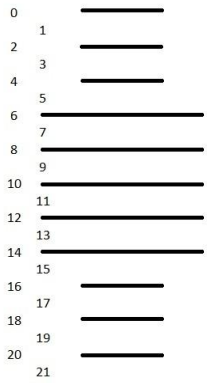
K chybnému určení výšky symbolu dochází převážně u složených not. Čára, která noty spojuje je detekována spolu s linkami, čímž ovlivní jejich detekci. Obrázek 5.23 zobrazuje takto ovlivněnou detekci linek. Většinou je poté detekováno šest linek a v tom případě, jak bylo řečeno, je jedna linka odstraněna. Pokud je odstraněna špatná linka, dojde k posunutí všech linek o jedna nahoru, či dolů, jak ukazuje Obrázek 5.24. Poté je posunuta i výška symbolu. Jde o problém značně ovlivňující výsledný zvuk, který se nepodařilo odstranit a stává se tak otázkou budoucí práce.



Obrázek 5.23: Detekce linek ovlivněna složenými notami.



Obrázek 5.24: Posunuté linky při určování výšky symbolů.



Obrázek 5.25: Podporované výšky symbolů.

Zpracování teček

Dalším krokem je zpracování teček, které prodlužují délku not a pomlk. Obrázek 5.18 ukazuje, že často dochází k chybné detekci teček. Takové tečky je potřeba odstranit. Také je potřeba detekovat symboly opakování. Veškeré tečky jsou při zpracování ze seznamu objektů smazány, aby nepřekážely při vytváření zvuku. Počet teček, patřících k notě či pomlce, je uložen přímo do struktury tohoto objektu. Jsou zkoumány vlastnosti teček na základě následujících podmínek (výšku symbolů zobrazuje Obrázek 5.25):

1. Výška tečky je 9 nebo 11 a následující symbol je tečka se stejnou x-ovou souřadnicí a výškou 9 nebo 11. Poté je tento symbol označen jako opakování.
2. Pokud je tečka prvním symbolem v osnově nebo je symbol před ní klíč, posuvka nebo taktová čára, je tečka pouze odstraněna.
3. Výška tečky je stejná jako výška symbolu před ní, nebo je jejich rozdíl jedna, pak je inkrementována proměnná určující počet teček u tohoto symbolu a tečka je odstraněna.
4. Pokud není splněna žádná z předchozích podmínek, je tečka odstraněna.

Zpracování posuvek

Posuvky mění výšku not a tím i výsledný zvuk, proto je nezbytné je zpracovat a modifikovat výšku ovlivněných not.

Nejprve je zpracováno předznamenání, tj. skupina béček, či křížků za notovým klíčem. Každá posuvka nacházející se v předznamenání modifikuje v celé osnově výšku not, jejichž aktuální výška je stejná jako výška posuvky, nebo je posunuta o násobky oktáv. Tj. jedna posuvka v předznamenání modifikuje výšku všech not na dané linkce, či mezeře ve všech oktávách. Je vytvořeno pole obsahující položku pro každou podporovanou výšku. Těch je 22 (Obrázek 5.25) a jsou inicializovány na hodnotu 0. Následuje průchod posuvkami v předznamenaní, tzn. jsou procházeny symboly následující notový klíč a pokud symbol není posuvka, zpracování předznamenání končí. Každá posuvka modifikuje hodnoty v poli pro každou oktávu. Pro béčko je uložena hodnota -1 a pro křížek hodnota 1.

Následuje detekce ostatních posuvek a modifikace výšky not. Je procházeno pole výšek a je z něj načtena aktuální hodnota posuvky (podle předznamenání). Pro každou výšku jsou procházeny všechny symboly a jsou zkoumány následující podmínky:

- Pokud je symbol taktová čára, aktuální hodnota posuvky je znovu načtena z předznamenání.
- Pokud je výška objektu stejná jako aktuální výška:
    - Pokud je objekt posuvka, je přepsána aktuální hodnota posuvky (béčko $\rightarrow -1$, křížek $\rightarrow 1$, odrážka $\rightarrow 0$).
    - Pokud je objekt nota, je upravena jeho výška na základě aktuální hodnoty posuvky a typu notového klíče. Tabulka 2 zobrazuje, jak je upravena výška jednotlivých not na základě aktuální hodnoty posuvky.

Typ Noty	Aktuální posuvka	
	Béčko	Křížek
G	+0,5	-0,5
F	+1,0	-0,5
E	+0,5	-1,0
D	+0,5	-0,5
C	+1,0	-0,5
B	+0,5	-1,0
A	+0,5	-0,5

Tabulka 2: Změna výšky noty na základě posuvky.
Tabulka ukazuje hodnoty, o které je výška změněna.

Kontrola počtu dob v taktu

Jak bylo zmíněno, post-processing v OMR aplikacích většinou obsahuje i kontrolu počtu dob v taktu, která má za úkol detekovat špatně klasifikované noty a opravit jejich typ. V aktuálním stavu této aplikace tato kontrola není možná z následujících důvodů:

- Je nutné vědět, kolik dob mají takty obsahovat, což je určeno časovou značkou. Bylo zamýšleno tuto značku detekovat, ovšem těchto značek může být velké množství, což by znamenalo mnoho tříd v modelu pro klasifikaci. Aktuálně jsou detekovány pouze některé základní časové značky. Navíc klasifikátor není stoprocentní a pokud dojde k chybné detekci časové značky, pak není možné kontrolu provést správně.

- K provedení kontroly počtu dob v taktu, musí být správně detekovány všechny taktové čáry. Stává se ovšem, že některé taktové čáry nejsou detekovány, což značně ztíží a v některých případech i znemožní provedení této kontroly.

Z těchto důvodu je kontrola počtu dob v taktu v aplikaci vynechána a stává se tak otázkou budoucí práce.

5.3.5 Tvorba a přehrávání zvuku

Nedílnou součástí aplikace je vytvoření a přehrání výsledného zvuku. Jelikož některé skladby mají hrát rychleji, jiné pomaleji, je uživateli umožněno zvolit délku celé noty (v rozmezí 0,1 až 4,0 sekundy), od které se odvíjí délky ostatních not. Uživatel ale většinou neví předem, jaké tempo by měla skladba mít, proto tento výběr probíhá až po zpracování obrazu. Z toho důvodu je zvuk vytvářen až po požadavku na přehrání či uložení skladby. V případě dlouhých skladeb a dlouhé celé noty trvá vytvoření zvuku nějakou dobu. Proto byla snaha tuto dobu zkrátit vytvořením zvuku ve vláknech pro každou osnovu zvlášť a poté jsou tyto zvuky spojeny do jednoho.

Vytvoření zvuku probíhá generováním vzorků pomocí funkce sinus. Frekvence vzorků byla zvolena 44100 Hz a byl zvolen stereo zvuk, tzn. dvoukanálový. Frekvence pro jednotlivé výšky jsou zapsány do hash tabulky. Následuje průchod objekty v osnově. Pro každou notu je načtena frekvence podle výšky a podle typu noty a počtu teček se určí délka noty. U mezer se určuje pouze délka, frekvence je nulová. Následuje generování vzorků, jejichž počet je odvozen od délky noty či mezery. Generování vzorků ukazuje Algoritmus 2. Lze vidět, že probíhá postupné zvyšování amplitudy na začátku zvuku a postupné snižování amplitudy na konci zvuku. To je provedeno z důvodu eliminace „klikání“ při přechodu mezi zvuky s odlišnou frekvencí. Vzorky jsou ukládány do pole bytů. Pole bytů získaná z jednotlivých osnov jsou spojena do jednoho pole.

Algoritmus 2 Generování vzorků zvuku jedné noty

$y = 0$;
$ramp = délka\ noty/20$;
For (; $y < ramp$; $y++$) { //postupné zvyšování amplitudy
    $vzorek =$
    $= (((sin(2 \cdot \pi \cdot (y + pozice\ ve\ výsledném\ zvuku)/(44100/frekvence))) \cdot 32767 \cdot y/ramp))$;
    přidat $vzorek$ do výsledného zvuku;
}
For (; $y < délka\ noty - ramp$; $y++$) { //maximální aplituda
    $vzorek =$
    $= (((sin(2 \cdot \pi \cdot (y + pozice\ ve\ výsledném\ zvuku)/(44100/frekvence))) \cdot 32767))$;
    přidat $vzorek$ do výsledného zvuku;
}
For (; $y < délka\ noty$; $y++$) { // postupné snižování amplitudy
    $vzorek =$
    $= (((sin(2 \cdot \pi \cdot (y + pozice\ ve\ výsledném\ zvuku)/(44100/frekvence))) \cdot 32767 \cdot$
    $(délka\ noty - y)/ramp))$;
    přidat $vzorek$ do výsledného zvuku;
}
$pozice\ ve\ výsledném\ zvuku\ += délka\ noty$;

Jelikož má zvuk dva kanály, je možné uložit a přehrát akordy, kde noty mají dvě hlavy. Samozřejmě existují akordy s více notovými hlavami, které ovšem v této aplikaci nejsou podporovány.

Přehrávání výsledného zvuku probíhá pomocí vestavěné třídy `AudioTrack`. Je vytvořena instance této třídy s potřebnými parametry, pomocí její metody `write()` je zapsán zvuk (zmíněné pole bytů) a pomocí další její metody `play()` je zvuk přehrán.

Uživatel má také možnost výsledný zvuk uložit. Byl zvolen zápis do wav souboru. V externí paměti je vytvořena složka *omr/savedSound*, kam jsou tyto soubory ukládány. Na Android není žádná knihovna, která by wav soubor vytvářela, tudíž je nutné ho vytvořit ručně. Nejprve je zapsána hlavička souboru, kterou ukazuje Tabulka 3 a za ni jsou zapsána data (vzorky v poli bytů).

Jméno	Délka v bajtech	Význam
wID	4	ASCII řetězec "WAVE".
fID	4	ASCII řetězec "frm ". Všechny ID bloků by měly obsahovat 4 znaky, proto ta mezera
fLen	4	Pevná hodnota, musí být vždy 16
wFormatTag	2	Tyto dva bajty vždy definují, jakým způsobem jsou vlastní zvuková data uložena. Většinou se setkáme s hodnotou 1, což znamená PCM (Pulse Code Modulation).
nChannels	2	Počet kanálů. Tudíž 1 = mono, 2 = stereo. Je možné mít i více než dva kanály, tyto případy nejsou však moc časté.
nSamplesPerSec	4	Kmitočet, uvádí se v Hz. Typické hodnoty jsou 11025 (telefonní kvalita), 22050 (hi-fi kvalita), 44100 (CD kvalita). Většinou se nesetkáme z hodnotamy nižšími než 8000 a vyššími než 48000.
nAvgBytesPerSec	4	Průměrný datový průtok za sekundu. Informace zejména pro přehrávače. U PCM formátu je tato hodnota však zbytečná, neboť si ji můžeme sami vypočítat vynásobením frekvence počtem kanálů a počtem bajtů na vzorek.
nBitsPerSample	2	Počet bitů na jeden vzorek. Toto čislo většinou nabýva hodnoty 8 nebo 16 bitů. Ačkoliv jsou osmibitové zvuky daleko méně kvalitnější, zabírají dvakrat méně místa, než šestnáctibitové.

Tabulka 3: Struktura hlavičky wav souboru.
Převzato z: http://www.fi.muni.cz/~qruzicka/Duffek/wav.html

5.4 Implementace

V této kapitole bude popsána stuktura výsledné aplikace, prostředí, ve kterém byla aplikace vyvíjena a použité knihovny.

5.4.1 Vývojové prostředí a použité knihovny

Aplikace byla vyvíjena v prostředí Eclipse s nádstavbou Android SDK. Programováno bylo v jazyce Java a byly použity knihovny OpenCV4Android SDK a jlibsvm (LIBSVM pro jazyk Java).

OpenCV je volně dostupná knihovna pro počítačové vidění a strojové učení, která obsahuje velké množství optimalizovaných algoritmů. Knihovna je dostupná pro operační systémy Windows, Linux, Mac OS a Android a obsahuje rozhraní pro programovací jazyky C, C++, Java, Python a MATLAB. [23]

LIBSVM je volně dostupná knihovna implementující klasifikaci metodou SVM. Tato knihovna poskytuje jednoduché rozhraní pro snadné použití v uživatelských programech a je dostupná pro celou řadu programovacích jazyků. [24]

5.4.2 Struktura aplikace

Strukturu aplikace zobrazuje Obrázek 5.26. Aplikace obsahuje hlavní třídu `MainActivity`, která rozšiřuje třídu `ActionBarActivity`. Třídy `ImagePreprocessing`, `StaffProcessing` a `SoundProcessing` rozšiřují třídu `AsyncTask`, která slouží pro spouštění operací na pozadí v tzv. pracovních vláknech a umožňuje publikovat výsledky v hlavním (UI) vlákně. Veškeré výpočty je potřeba provádět právě v pracovních vláknech, aby nedošlo k zamrznutí uživatelského rozhraní. Je spouštěno jedno vlákno pro předzpracování obrazu. Pro zpracování osnov a vytvoření zvuku je spuštěno tolik vláken, kolik je nalezeno notových osnov v obraze.

Hlavní třída `MainActivity` se stará o zobrazování jednotlivých obrazovek aplikace a o spouštění vláken zpracovávajících obraz a zvuk. Po zpracování obrazu tato třída obsluhuje požadavky na přehrání a uložení hudby.

Třída `ImagePreprocessing` se stará o předzpracování obrazu. Obsahuje funkce pro změnu velikosti obrazu, binarizaci a nalezení notových osnov. Dále jsou zde načteny modely pro klasifikaci.

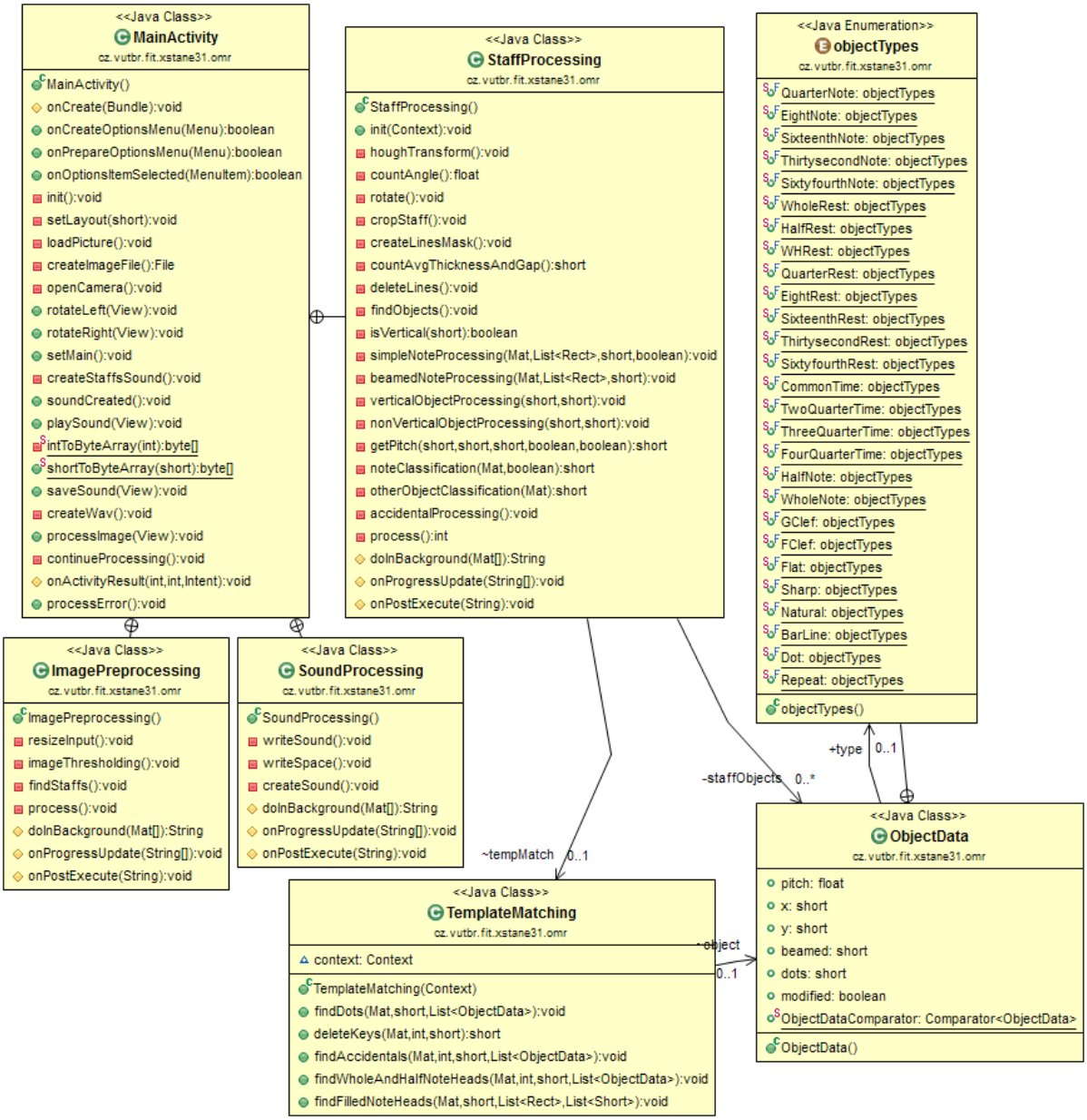
Třída `StaffProcessing` se stará o zpracování jedné osnovy. Obsahuje funkce potřebné pro odstranění linek a detekci a zpracování objektů. Nalezené a zpracované objekty předává hlavní třídě, kde je z nich poté vytvořen zvuk.

Třída `SoundProcessing` se stará o vytvoření zvuku ze zpracovaných objektů pro jednu osnovu. Generuje vzorky pomocí funkce sinus na základě frekvence not. Vzorky ukládá do pole bytů, které předává hlavní třídě, kde jsou poté spojena tato pole z jednotlivých osnov.

Třída `TemplateMatching` obsahuje funkce implementující detekci objektů pomocí metody template matching.

Důležitou třídou je `ObjectData`, která slouží jako struktura pro uložení objektu. Pro každý objekt nalezený v osnově je vytvořena entita této třídy a jsou nastaveny její atributy podle typu objektu a pozice v osnově. Třída obsahuje výčet všech podporovaných typů objektu (`objectTypes`). Atributy a jejich význam je následující:

- *type*: obsahuje typ objektu
- *pitch*: obsahuje výšku objektu
- *x, y*: pozice objektu v osnově
- *beamed*: číslo složené noty – je uloženo pouze u not, které jsou částmi složené noty, u ostatních objektů je hodnota tohoto atributu -1
- *dots*: počet teček patřících k symbolu
- *modified*: hodnota tohoto atributu je true, pokud byla změněna výška symbolu při zpracování posuvek (je využit k tomu, aby nebyla výška modifikována vícekrát)



Obrázek 5.26: Diagram tříd výsledné aplikace.

6 Testování a vyhodnocení

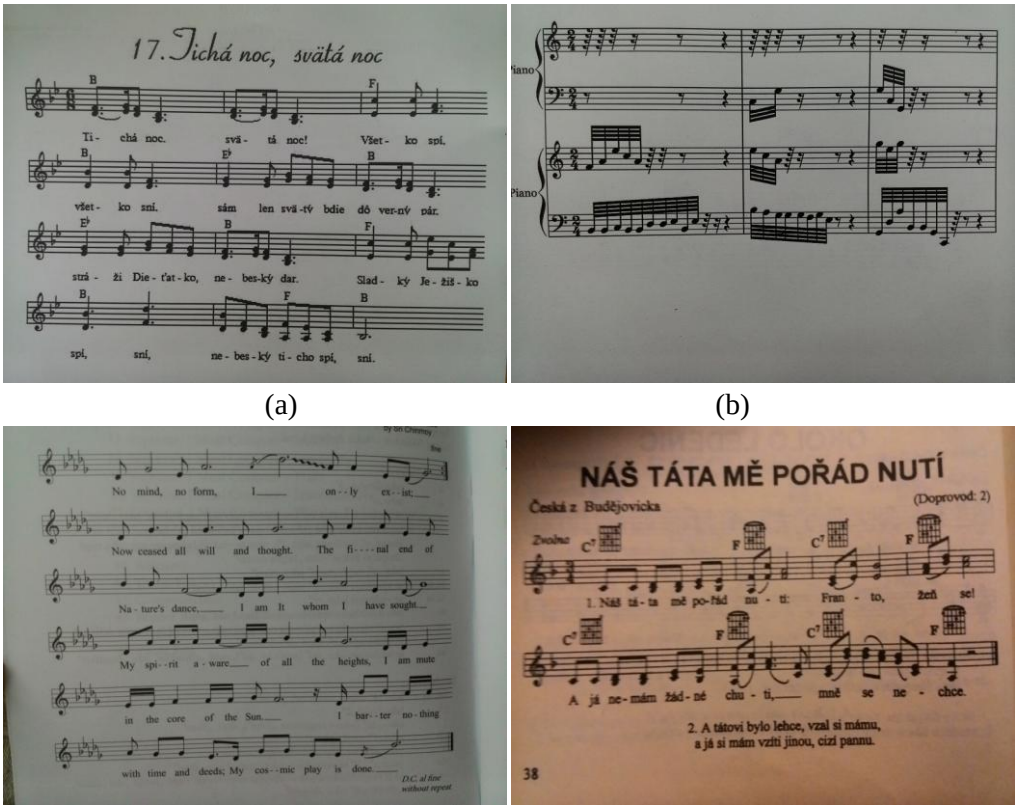
V této kapitole bude popsáno vyhodnocení výsledné aplikace. Nejprve bude ukázáno, jakým způsobem a na jaké sadě obrázků byla aplikace testována. Dále bude uvedeno vyhodnocení důležitých částí aplikace, kterými jsou klasifikace a doba zpracování obrazu.

6.1 Testovací sada a použitá zařízení

Aplikace byla vyvíjena a testována na telefonu Asus Zenfone 4 A450CG, jehož parametry ukazuje Tabulka 4. Pomocí tohoto telefonu byla pořízena sada, obsahující 101 fotografií. Některé fotografie byly pořízeny v rozlišení 5 mpx, jiné 8 mpx. Ukázku fotografií zobrazuje Obrázek 6.1.

Asus Zenfone 4 A450CG	
Operační systém	Android 4.4 (KitKat)
CPU	Intel Atom Z2520 (1,2 Ghz)
Paměť	1 GB LPDDR2 RAM
Fotoaparát	8 Mega-Pixel, F2.0 Aperture, PixelMaster

Tabulka 4: Parametry telefonu Asus Zenfone 4 A450CG.



(a) (b)

(c) (d)

Obrázek 6.1: Ukázka obrázků z testovací sady.

Obrázky byly foceny nejprve ze zpěvníků. Poté, při tvorbě datasetu, bylo potřeba získat další obrázky, obsahující symboly, které se v běžných písních nevyskytují. Byly vygenerovány notové zápisy, které byly poté vytisknuty a vyfoceny. Obrázek 6.1 (b) zobrazuje příklad takového obrázku. V testovací sadě jsou obsaženy jak obrázky kvalitní (Obrázek 6.1 (a), (b)), tak i obrázky nekvalitní, tzn. rozmazané, či geometricky deformované (Obrázek 6.1 (c), (d)).

Testování probíhalo následujícím způsobem. Každý obrázek byl zpracován aplikací. Ta po zpracování vypsala seznam nalezených objektů ve tvaru:
`[výška, typ objektu, x-ová souřadnice, y-ová souřadnice, složená nota, počet teček]`.
Příklad výpisu zobrazuje Obrázek 6.2. Tyto výpisy byly poté ručně porovnány s příslušnými obrázky a veškeré nepřesnosti byly označeny a zpracovány. Výsledky budou ukázany v následujících kapitolách.

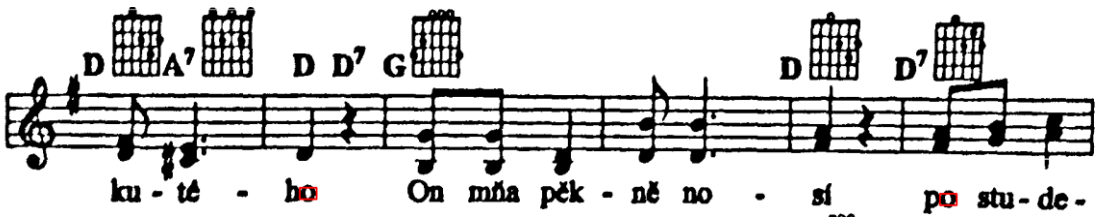
```
1
[0.0, GClef, 0, 0, -1, 0]
[6.0, Sharp, 175, 77, -1, 0]
[15.0, QuarterNote, 289, 186, -1, 1]
[14.0, EightNote, 397, 175, -1, 0]
[15.0, QuarterNote, 464, 189, -1, 0]
[0.0, BarLine, 526, 0, -1, 0]
[17.0, HalfNote, 578, 211, -1, 1]
[0.0, BarLine, 722, 0, -1, 0]
[15.0, QuarterNote, 774, 192, -1, 1]
[14.0, EightNote, 881, 181, -1, 0]
[15.0, QuarterNote, 965, 194, -1, 0]
[0.0, BarLine, 1026, 0, -1, 0]
[17.0, HalfNote, 1075, 215, -1, 1]
[0.0, BarLine, 1217, 0, -1, 0]
[11.0, QuarterNote, 1409, 151, -1, 0]
[0.0, BarLine, 1468, 0, -1, 0]
[0.0, BarLine, 1668, 0, -1, 0]
[12.0, HalfNote, 1718, 161, -1, 0]
[12.0, QuarterNote, 1855, 164, -1, 0]
[0.0, BarLine, 1919, 0, -1, 0]
04-23 16:08:33.851: I/setMain(5298): 1: 3
```

Obrázek 6.2: Výpis objektů nalezených aplikací v jedné osnově.

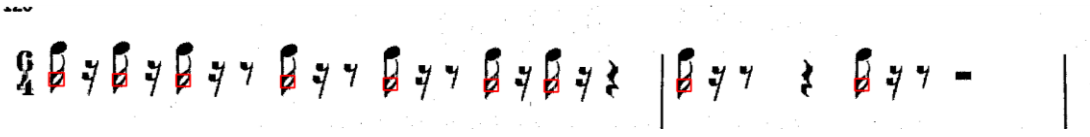
6.2 Vyhodnocení metody template matching

U metody template matching bylo vyhodnoceno, zda je objekt detekován správně, zda není detekován vůbec či zda je detekován někde, kde se nevyskytuje. Tabulka 5 zobrazuje výsledky tohoto vyhodnocení. Lze pozorovat několik nedostatků, kterými jsou chybná detekce celých a půlových not a nedetekované tečky a posuvky.

Celé a půlové noty jsou často detekovány tam, kde se nevyskytují. U celých not, je to většinou v textu písně, jelikož některá písmena (většinou je to písmeno „o“) jsou tvarem velmi podobná celé notě. Příklad chybně detekovaných celých not zobrazuje Obrázek 6.3. U půlových not dochází k chybné detekci převážne v praporu šestnáctinových a dvaatřicetinových not, jak zobrazuje Obrázek 6.4. K chybné detekci celých a půlových not dochází kvůli použitým vzorům. Vzory jsou ovšem vytvořeny a použity na základě mnohých testování tak, aby byl poměr *doba zpracování/úspěšnost* co nejlepší. V případě potřeby vyšší úspěšnosti by bylo nutné vytvořit více lepších vzorů.



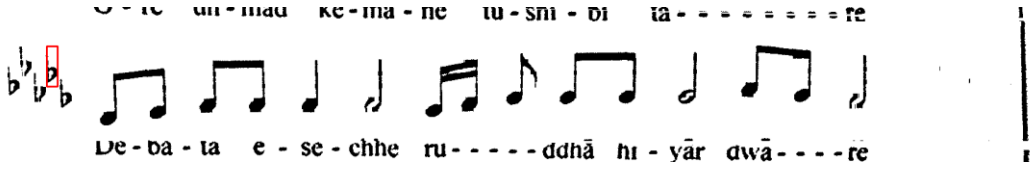
Obrázek 6.3: Chybně detekované celé noty (označeny červenými obdélníky).



Obrázek 6.4: Chybně detekované půlové noty (označeny červenými obdélníky).

Dalším problémem jsou nedetekované tečky. Tečky pro tuto práci nejsou příliš důležité, jelikož „pouze“ prodlužují délku not a pomlk, což nemá tak velký vliv na výsledný zvuk, jako například chybějící noty či posuvky. Proto byl použit pouze jeden vzor, aby byla ušetřena výpočetní doba. Nedetekované tečky ovšem ovlivní detekci symbolů pro opakování. Jelikož v aktuálním stavu aplikace není brán na opakování zřetel, nebylo potřeba toto řešit. V budoucnu by bylo opět nutné vytvořit více vzorů pro detekci teček.

Často dochází k nedetekování posuvek. Chybějící posuvky již mají větší vliv na výsledný zvuk a proto zde byla snaha dosáhnout co nejlepší úspěšnosti. Nejčastěji nejsou posuvky detekovány v předznamenání (Obrázek 6.5). Je to hlavně z toho důvodu, že obraz je zde deformován (při focení ze zpěvníku je stránka ohnutá) či špatně osvětlen. Proto při zpracování (binarizace, odstranění linek) dojde k rozbití těchto symbolů, načež takové symboly nejsou detekovány. Metoda template matching se ukázala být velmi citlivá na veškeré odchylky od vzorů. Proto pro rozbité či mírně nakloněné objekty je potřeba vytvořit další vzory. Aktuálně se aplikace soustředí na úspěšné zpracování kvalitně pořízených fotografií, proto na těch méně kvalitních stále dochází k nedetekování posuvek. Odstranění tohoto jevu je otázkou budoucí práce.



Obrázek 6.5: Nedetekované posuvky. Obrázek ukazuje 4 nedetekovaná béčka.

Metodou template matching jsou také detekovány plné notové hlavy při zpracování not (viz kapitola Segmentace 5.3.2). Také tato detekce se neobejde bez chyb a často dochází k tomu, že hlavy nejsou detekovány. To vede k tomu, že nota není zpracována a ve výsledku chybí. Počet takto nedetekovaných not je poměrně vysoký (viz Tabulka 6). Čísla uvedená v této tabulce obsahují i noty, které nebyly označeny jako vertikální z důvodu rozbití notové nohy při odstranění linek. Tento počet je ale malý v poměru k nedetekovaným hlavám. K nedetokování notových hlav dochází opět z důvodu nekvalitního vstupního obrazu a následného špatného zpracování. I v tomto případě by bylo potřeba vytvořit více vzorů.

Navíc, jak bylo zmíněno opět v sekci segmentace, dochází k chybné detekci hlav. I přes snahu tyto hlavy odstranit, zůstalo v použité sadě obrázků stále 39 chybně detekovaných hlav, které jsou poté zpracovány a přehrány.

Typ objektu	Detekováno	Nedetekováno	Chybně
Celé noty	52 (95%)	3 (5%)	16
Půlové noty	213 (97%)	6 (3%)	118
Tečky	197 (75%)	65 (25%)	0
Opakování	42 (66%)	22 (34%)	6
Béčka	508 (80%)	129 (20%)	0
Křížky	100 (88%)	13 (12%)	0
Odrážky	64 (88%)	9 (12%)	0
G klíč	270 (99%)	2 (1%)	0
F klíč	122 (99%)	1 (1%)	0

Tabulka 5: Detekce objektů metodou template matching.

6.3 Vyhodnocení klasifikace metodou SVM

V této části je vyhodnocena úspěšnost klasifikace metodou SVM. Touto metodou jsou klasifikovány noty čtvrťové až čtyřiašedesátinové a pomlky celé až čtyřiašedesátinové. Jsou klasifikovány i některé

časové značky, ale jelikož nejsou dále použity, tak nejsou zahrnuty do vyhodnocení. Vyhodnocení je rozděleno na dvě části. V první je vyhodnocena úspěšnost klasifikace not a ve druhé úspěšnost klasifikace pomlk.

Výsledek klasifikace not zobrazuje Tabulka 6. Tato tabulka ukazuje, jaký typ byl jednotlivým notám přidělen klasifikátorem. Řádky označují noty, vyskytující se v notových zápisech a sloupce označují typ, který byl notám přiřazen. Například v prvním řádku jsou čtvrťové noty. Celkem se jich v notových zápisech objevilo 1269, z toho 1165 bylo klasifikováno správně (tzn. jako čtvrťová nota), 4 byly klasifikovány jako osminová nota a 5 jako šestnáctinová nota. U ostatních typů not dochází k chybné klasifikaci častěji. Je to dáno nedokonalým datasetem, jehož tvorba byla popsána v kapitole Klasifikace 5.3.3. Problémem je nedostatek obrázků u tříd reprezentujících symboly, které se v běžných notových zápisech často nevyskytují. Dále by bylo potřeba některé třídy rozdělit do více tříd, jelikož obrázky v několika třídách jsou dosti odlišné (viz Obrázek 6.6).

Obrázek 6.6: Ukázka obrázků z datasetu pro osminové noty.

Dále lze vidět, že často dochází k nedetekování not. Jak bylo zmíněno v předchozí sekci, je to dáno špatným zpracováním nekvalitních vstupních obrazů a nedetekováním hlav metodou template matching.

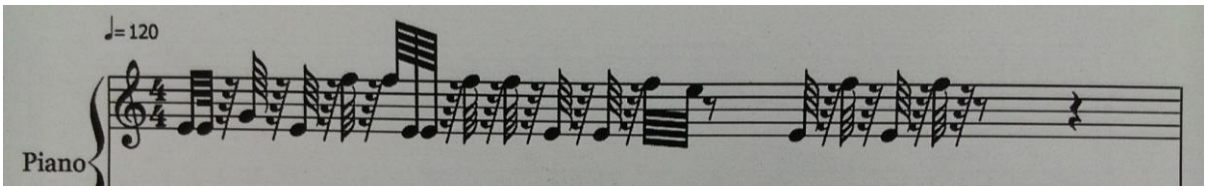
NOTY		Detekováno jako					Nedetekováno	Celkem
		1/4	1/8	1/16	1/32	1/64		
Typ noty	1/4	**1165 (92%)**	4	5	0	0	95 (7%)	1269
	1/8	10	**1562 (87%)**	102	4	8	100 (6%)	1786
	1/16	1	27	**545 (90%)**	12	11	7 (1%)	603
	1/32	0	14	57	**469 (71%)**	88	31 (5%)	659
	1/64	0	22	23	33	**466 (75%)**	77 (12%)	621

Tabulka 6: Klasifikace not pomocí SVM.

U pomlk se situace ukázala být mnohem lepší a pokud byla pomlka detekována, klasifikátor neměl problém s jejím rozpoznáním. Výsledek klasifikace pomlk zobrazuje Tabulka 7. Lze vidět, že pouze jednou došlo k chybné klasifikaci, kdy čtyřiašedesátinová pomlka byla klasifikována jako dvaatřicetinová.

Problémem je opět nedetekování pomlk. Jak bylo zmíněno v sekci Segmentace 5.3.2, může se stát, že je pomlka během zpracování rozbita a neprojde až ke klasifikaci. Dalším problémem je, že někdy je čtvrťová pomlka označena jako vertikální symbol, načež je zpracována jako nota. Většinou je poté označena jako taktová čára, nebo není vůbec uložena do výsledku, jelikož neobsahuje notovou hlavu a je příliš široká na to, aby byla označena za taktovou čáru.

V tabulce je vidět opravdu velké množství nedetekovaných čtyřiašedesátinových pomlk. To je ovšem způsobeno špatně vygenerovanými obrázky (viz Obrázek 6.7), kdy nelze oddělit pomlku od noty a není to chybou programu.



Obrázek 6.7: Špatně vygenerovaná osnova. Pomlky nelze oddělit od not.

POMLKY		Správně	Nedetekováno	Chybně
Typ pomlky	1/1	39 (95%)	2 (5%)	0
	1/2	54 (96%)	2 (4%)	0
	1/4	180 (90%)	19 (10%)	0
	1/8	265 (96%)	10 (4%)	0
	1/16	425 (98%)	9 (2%)	0
	1/32	410 (96%)	16 (4%)	0
	1/64	287 (86%)	45 (14%)	1

Tabulka 7: Klasifikace pomlk pomocí SVM.

6.4 Doba zpracování obrazu

Důležitou vlastností aplikace je běh v reálném čase, proto byla otestována doba zpracování obrazu na různých zařízeních. Byly měřeny dva časové údaje:

- Průměrná doba předzpracování obrazu, tj. binarizace, detekce osnov a načtení modelů pro klasifikaci.
- Průměrná doba zpracování jedné osnovy, tj. doba běhu jednoho vlákna.

Tato doba byla měřena při zpracování fotografie, kterou zobrazuje Obrázek 5.2.

Tabulka 8 ukazuje naměřené hodnoty. Jak lze vidět, na novějších výkonných zařízeních je doba zpracování celého obrázku kolem jedné minuty, na méně výkonných zařízeních je potřeba počítat s dobou zpracování dosahující až čtyř minut. Je nutné ovšem říct, že na všech zmíněných zařízeních je možno spustit pouze 5 souběžně pracujících vláken. Proto, pokud notový zápis obsahuje více než 5 notových osnov, je potřeba počítat s delší dobou zpracování, jelikož se čeká na dokončení některého z běžících vláken a až poté je spuštěno zpracování další osnovy.

Během testování se ukázalo, že aplikace je paměťově velice náročná, proto je potřeba zařízení s alespoň 1GB RAM. Bylo testováno i na zařízeních s menší pamětí, na kterých ovšem aplikace právě kvůli nedostatku paměti padala.

Zařízení	Doba předzpracování obrazu (s)	Doba zpracování jedné osnovy (s)
Sony Xperia Z3 compact	11	31
Samsung Galaxy S4	15	46
Dell Venue 7	20	60
Sony Xperia M2 Aqua	35	56
Nexus 7	30	67
Sony Xperia L	53	107
HTC Evo 3D	57	144
Asus Zenfone 4 A450CG	51	160

Tabulka 8: Doba zpracování obrazu na různých zařízeních.

Závěr

V rámci této práce byla vytvořena aplikace pro automatické rozpoznávání a přehrávání not z fotografií (dále OMR). Tato aplikace je určena pro mobilní zařízení s operačním systémem Android. Umožňuje zpracovat fotografii pořízenou fotoaparátem nebo načtenou z paměti zařízení. Po zpracování uživatel může výsledný zvuk přehrát nebo jej uložit.

Byla nastudována existující řešení tohoto problému od těch nejstarších až po současná. Dále byly nastudovány a popsány algoritmy umožňující rozpoznání notového zápisu z fotografie.

Hned na počátku vývoje aplikace se ukázalo, že obrázky pořízené fotoaparátem mobilního zařízení nelze kvalitou srovnávat s obrázky naskenovanými, které jsou vstupem existujících řešení. Proto nebylo možné použít jednoduché metody pro detekci notových osnov a notových linek, jako je použití pouze horizontální projekce a byly hledány robustnější metody. Velká síla se ukála být v morfologických operacích, které se v poslední době dostávají do popředí právě při řešení problému OMR. Ovšem ani morfologické operace nejsou všemocné a pokud je na vstupu nekvalitní obrázek (viz Obrázek 5.1), nelze očekávat dobrý výsledek. Aplikace tedy byla vyvíjena s cílem úspěšně zpracovat kvalitní vstupní obrázky (viz Obrázek 5.2). Byla ale zároveň snaha co nejlépe zpracovat i ty méně kvalitní.

Výsledná aplikace se potýká s několika problémy. První z nich se vyskytuje u již zmíněných nekvalitních vstupních obrazů, kdy může dojít k chybné detekci notových linek, načež jsou špatně odstraněny, což znemožní správnou detekci objektů. Dalším problémem zmíněným v kapitole 5.3.4 je chybné určení výšky symbolů (vyskytující se převážně u složených not), což negativně ovlivní výsledný zvuk. Další problémy se vyskytují u klasifikace objektů.

Pro klasifikaci byly použity dvě metody: template matching a SVM. Metoda template matching se ukázala být velmi citlivá na věškeré odchylky od vzorů. Jelikož byla snaha dosáhnout co nejlepšího poměru *doba zpracování/úspěšnost*, nemohlo být použito velké množství vzorů a výsledek této metody není ideální, jak ukazuje Tabulka 5.

Pro metodu SVM bylo potřeba vytvořit vlastní dataset. Klasifikace pomlk se ukázala být velmi úspěšná a k chybné klasifikaci došlo pouze jednou, jak ukazuje Tabulka 7. U not je situace poněkud horší a k chybné klasifikaci dochází často (viz Tabulka 6).

Budoucí práce by měla spočívat v odstranění zmíněných nedostatků. V případě chybné detekce a odstranění linek by bylo potřeba nějakým způsobem vylepšit vstupní obraz. To ale není cílem takovéto aplikace. Navíc použitá metoda založená na morfologických operacích je dostatečně robustní a poradí si i s mírnou geometrickou deformací obrazu a pokřivením linek. Proto je na uživateli, aby pořídil co nejlepší obraz.

U metody template matching by bylo potřeba vytvořit mnohem více vzorů. Otázkou pak ale bude doba zpracování obrazu. Tabulka 8 ukazuje dobu zpracování na různých zařízeních. Na novějších výkonných zařízeních zpracování obrazu proběhne rychle a například dvojnásobná doba by neuškodila, pokud by došlo k výraznému vylepšení výsledku. Ovšem je potřeba brát v potaz i méně výkonná zařízení, na kterých již teď dosahuje doba zpracování až čtyř minut. Ale jelikož jakákoliv chyba při použití této metody značně ovlivní výsledný zvuk, bude potřeba se v první řadě zaměřit na vylepšení této části, čímž bohužel dojde k prodloužení doby zpracování obrazu.

Metoda SVM je silně závislá na použitém datasetu. Jak bylo řečeno, pro pomlky je tento dataset ideální. Pro noty by bylo potřeba jej přepracovat.

Literatura

[1] Bainbridge D., Bell T.: The Challenge of Optical Music Recognition, Computers and the Humanities, roč. 35, 2001, s. 95-121

[2] Bellini P., Bruno I., Nesi P.: Optical music sheet segmentation, Proceedings First International Conference on WEB Delivering of Music, Nov. 2001, s. 183-190

[3] Bloch I., Rossant F.: A fuzzy model for optical recognition of musical scores, Fuzzy Sets and Systems, 2004, roč. 141, č. 2, s. 165-201

[4] Capela A., Cardoso J. S., Carrapatoso E., Da Costa J. F. P., Guedes C., Rebelo A.: A Shortest Path Approach for Staff Line Detection, Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, Nov. 2007, s. 79-85

[5] Capela A., Cardoso J. S., Rebelo A., Guedes C.: A connected path approach for staff detection on a music score, 2008 15th IEEE International Conference on Image Processing, Oct. 2008, s. 1005-1008

[6] Capela A., Cardoso J. S., Rebelo A.: Optical recognition of music symbols, International Journal on Document Analysis and Recognition (IJDAR), 2010, roč. 13, č. 1, s. 19-31

[7] Cottrell G. W., Kanan C.: Color-to-Grayscale: Does the Method Matter in Image Recognition?, PloS ONE, 2012, roč. 7, č. 1, s. e29740

[8] Dalal N., Triggs B.: Histograms of Oriented Gradients for Human Detection, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, roč. 1, s. 886-893

[9] Fokkinga M.: The Hough transform, Journal of Functional Programming, 2011, roč. 21, č. 2, s. 129-133

[10] Fornés A., Lladós J.: A Symbol-Dependent Writer Identification Approach in Old Handwritten Music Scores, Frontiers in Hadwriting Recognition, Nov. 2010, s. 634-639

[11] Forsyth D. A., Ponce J.: Computer vision a modern approach, New Jersey: Prentice-Hall, 2003, 693 s., ISBN 0-13-191193-7

[12] Fujinaga I.: Adaptive Optical Music Recognition, Montréal, 1996, Doktorská dizertační práce, McGill University, Faculty of Music

[13] Hsin-Hua Chen, Mu-Chun Su, Wan-Chi Cheng: A Neural-Network-Based Approach to Optical Symbol Recognition, Neural Processing Letters, roč. 15, 2002, s. 117-135

[14] Jain R., Kasturi R., Schunck B. G.: Machine vision, Boston: McGraw-Hill, 1995, 549 s., ISBN 0-07-032018-7

[15] Lu S., Pal U., Su B., Tan C. L.: An Effective Staff Detection and Removal Technique for Musical Documents, 2012 10th IAPR International Workshop on Document Analysis Systems, March 2012, s. 160-164

[16] Rossant F.: A global method for music symbol recognition in typeset music sheets, Pattern Recognition Letters, roč. 23, č. 10, 2002, s. 1129-1141

[17] Russ J. C.: The image processing handbook, 4th ed., Florida: CRC Press LLC, 2002, 732 s., ISBN 0-8493-1142-X

[18] Shapiro L. G., Stockman G. C.: Computer vision, New Jersey: Prentice-Hall, 2001, 580 s., ISBN 0-13-030796-3

[19] Soille P.: Morphological image analysis: principles and applications, 2nd ed., Berlin: Springer-Verlag, 2003, 391 s., ISBN 3-540-42988-3

[20] Sonka M., Hlavac V., Boyle R.: Image Processing, Analysis, and Machine Vision, 3rd ed., Toronto: Thomson, 2008, 829 s., ISBN-10 0-495-08252-X, ISBN-13 978-0-495-08252-1

[21] Tajeripour F., Sotoodeh M.: A novel staff removal method for printed music image, IEICE Electronics Express, roč. 9, č. 7, 2012, s. 609-615

[22] Gehrkens K. W.:Music notation and terminology [online], 2006-11-08, aktualizováno 2011-02-14 [cit. 2014-11-25], Dostupné na URL: http://www.gutenberg.org/files/19499/19499-h/19499-h.htm

[23] OpenCV (Open source computer vision) [online], [cit. 2015-05-17], Dostupné na URL: http://opencv.org/

[24] LIBSVM -- A Library for Support Vector Machines [online], [cit. 2015-05-17], Dostupné na URL: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Seznam příloh

Příloha 1. DVD

Příloha 1. DVD

Přiložené DVD obsahuje:

- doc/ Dokumentace vygenerovaná pomocí JavaDoc.
- images/ Sada obrázků použitých při testování.
- src/ Zdrojové kódy aplikace.
- zpráva/ Zpráva ve formátu pdf a docx.
- OMR.apk Instalační balík aplikace.