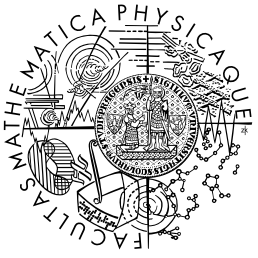


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Kristýna Harvanová

Postprocessing syntetických notopisů v kontextu jejich rozpoznávání

Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Jiří Mayer
Studijní program: Informatika

Praha 2024

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Děkuji panu Mgr. Jiřímu Mayerovi za odborné vedení, pomoc a cenné rady, které mi poskytl při zpracování bakalářské práce. Ráda bych poděkovala také panu MgA. et Mgr. Jan Hajič, jr., Ph.D. za příležitost aktivně se účastnit setkání Prague Music Computing Group a sdílet znalosti z oblasti Optical Music Recognition. Děkuji mým profesorům na Konzervatoři Jaroslava Ježka za ohleduplnost a vstřícnost v období psaní mé bakalářské práce.

Název práce: Postprocessing syntetických notopisů v kontextu jejich rozpoznávání

Autor: Kristýna Harvanová

Katedra: Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Jiří Mayer, Ústav formální a aplikované lingvistiky

Abstrakt: Tato práce se zaměřuje na vylepšení metod syntézy trénovacích dat pro úlohu Optical Music Recognition (OMR). Práce se soustředí na vytváření realistických barevných a degradovaných obrázků not (postprocessing). Tato degradovaná data vznikají ze syntetických čistě černo-bílých obrázků. Po aplikaci postprocessingových metod notopisy věrně napodobují fyzické dokumenty, čímž zlepšují kvalitu trénovacích dat pro OMR modely. Navržené postprocessingové metody byly testovány na úloze object detection, neboli na rozpoznávání jednotlivých typů různých hudebních symbolů. Experimenty prokázaly, že všechny navržené metody pozitivně ovlivňují výsledný OMR model, přičemž největší přínos má generování syntetického pozadí notopisů.

Klíčová slova: optické rozpoznávání notopisů, syntéza dat, hluboké učení

Title: Postprocessing of Synthetic Sheet Music in the Context of Optical Music Recognition

Author: Kristýna Harvanová

Department: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Jiří Mayer, Institute of Formal and Applied Linguistics

Abstract: This work focuses on improving training data synthesis methods for the Optical Music Recognition (OMR) task. The study concentrates on creating realistic, colored, and degraded images of musical scores (postprocessing). These degraded data are generated from synthetic, purely black-and-white images. After applying postprocessing methods, the musical scores closely mimic physical documents, thereby enhancing the quality of training data for OMR models. The proposed postprocessing methods were tested on object detection tasks, specifically recognizing various types of musical symbols. Experiments demonstrated that all proposed methods positively impact the resulting OMR model, with the greatest benefit coming from the generation of synthetic backgrounds for musical scores.

Keywords: optical music recognition, data synthesis, deep learning

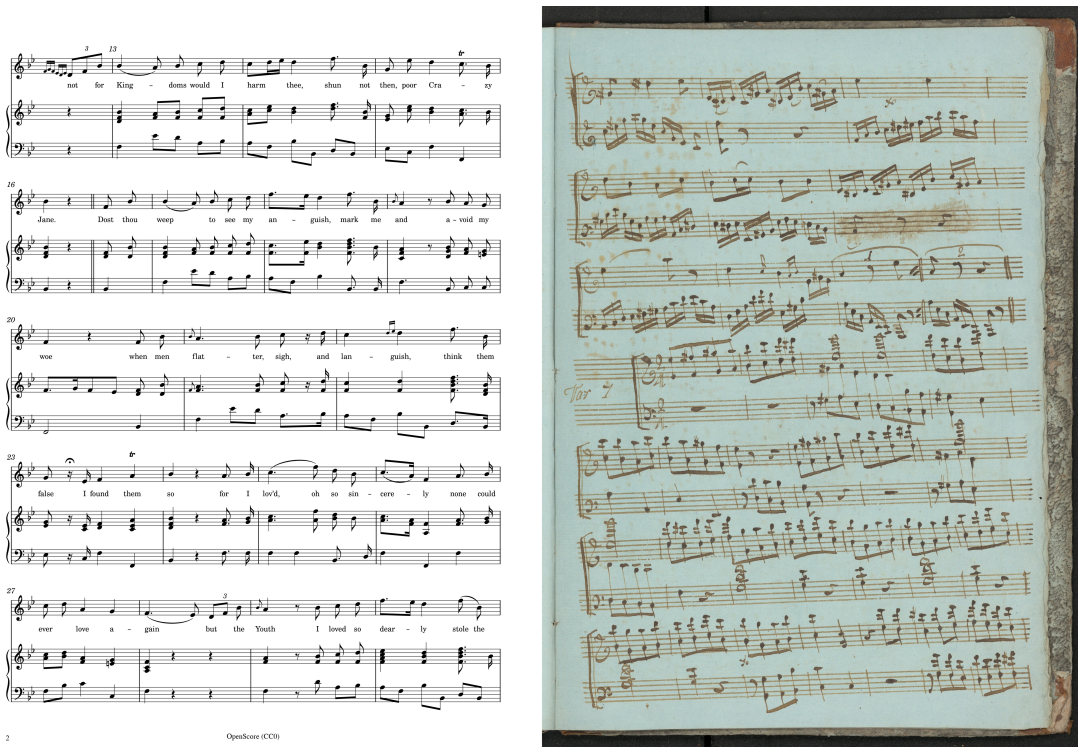
Obsah

1 Úvod

Optické rozpoznávání hudební notace (tzv. notopisů), známé jako *Optical Music Recognition* (OMR), je proces převádění obrázků not na strojově čitelný formát [1]. Tento proces je podobný úloze optického rozpoznávání textu, kde je cílem zisk informací v textu z fotografií nebo skenů dokumentů. Nejznámější podúlohou optického rozpoznávání textu je *Optical Character Recognition* (OCR) neboli rozeznávání jednotlivých znaků.

Úloha OMR zahrnuje několik podúloh, jako například detekci linek osnov, analýzu obsahu či struktury stránky, detekci notových hlaviček a jiných hudebních symbolů, nebo i samotné pochopení hudebního obsahu stránky. Oproti OCR či jiným úlohám rozpoznávání textu se tak jedná o úlohu výrazně složitější, jak mimo jiné uvádí Calvo-Zaragoza, Hajič a Pacha ve svém článku [1]. Výsledky OMR lze využít pro různé aplikace, například vyhledávání podle melodie, přehrání hudby, editaci, digitální tisk či muzikologickou analýzu.

Pro účely této práce rozlišujeme mezi dvěmi hlavními kategoriemi notopisů. Jedná se o ručně psané notopisy a o tištěné noty, což jsou notopisy tvořeny zpravidla notačními programy. Jejich vzhled se významně liší, jak ukazuje přiložený Obrázek 1.1.



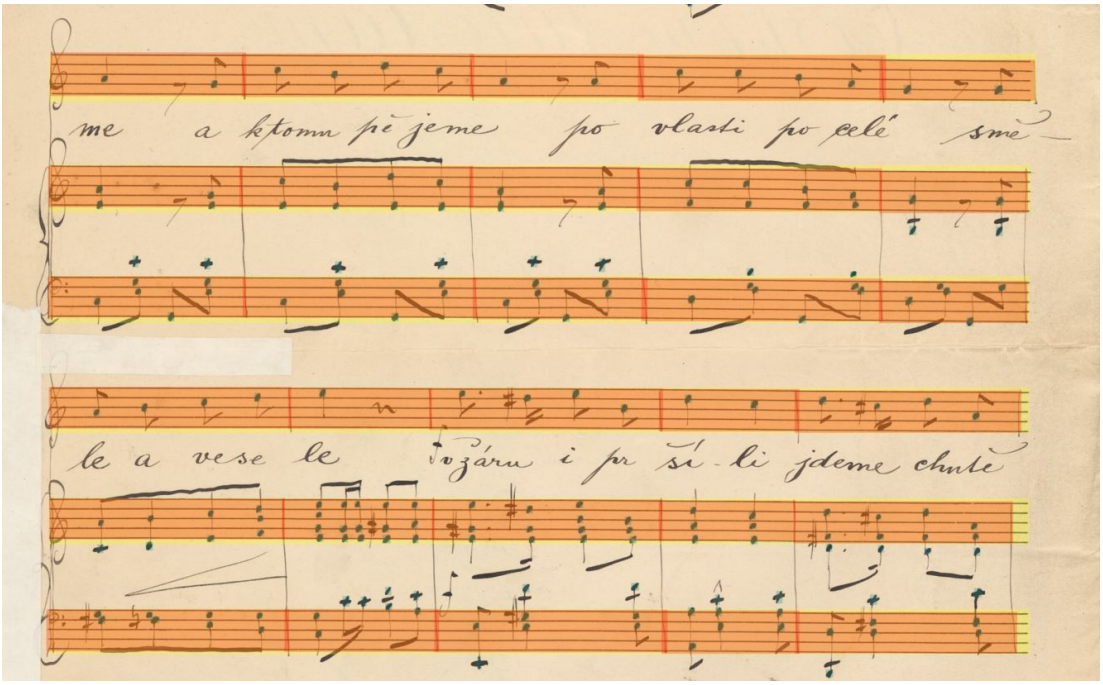
Obrázek 1.1 Tištěný notopis (vlevo). Sken ručně psaného notopisu (vpravo). Ručně psané notopisy se vyznačují větší pestrostí jak mezi autory, tak i v rámci jedné stránky. Jejich rozpoznávání tak představuje výrazně těžší úlohu.

Dosud nejúspěšnějším přístupem k řešení OMR je využití metod hlubokého učení. K jejich trénování jsou však potřeba trénovací data, která v případě OMR problému sestávají z obrázků notopisů a anotací jejich hudebního obsahu (notové hlavičky, klíče, notová osnova aj.). Získání těchto oanotovaných dat je však příliš

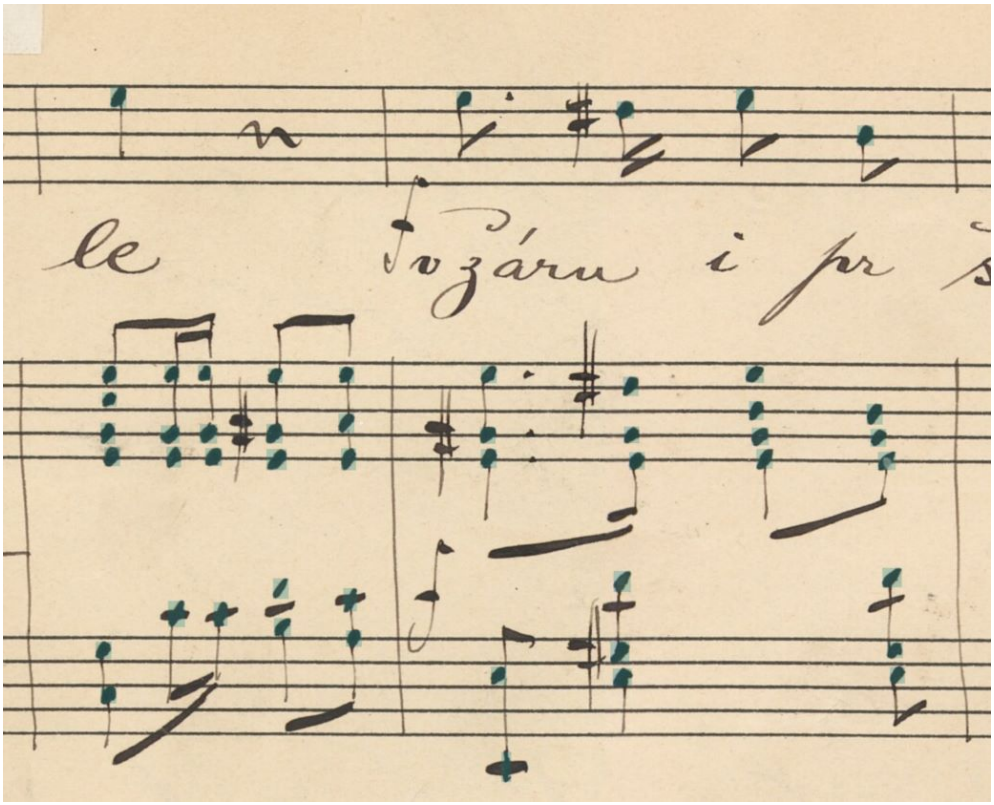
drahé a časově náročné, neboť je nutno anotovat data ručně. Proto se velmi často přistupuje k tzv. syntéze trénovacích dat [2], [3], [4]. Oblast syntézy trénovacích dat v kontextu OMR bude hlavním obsahem této práce.

Současné syntezátory trénovacích dat pro OMR se obvykle zaměřují na hudební obsah obrázku a tak často produkují pouze černo-bílé a dokonale zarovnané obrázky, podobně jako nástroje na profesionální tisk not. Tato práce se snaží současné výstupy syntézy notopisů vylepšit takovým způsobem, že z černo-bílých obrázků se snaží vytvořit barevné a degradované. Zkrátka takové, aby vypadaly jako sken nebo fotografie fyzického používaného dokumentu. Cílem práce není měnit hudební obsah obrázku, ale zůstat v obrazové doméně a zavést a ověřit řadu metod pro degradace černo-bílých obrázků.

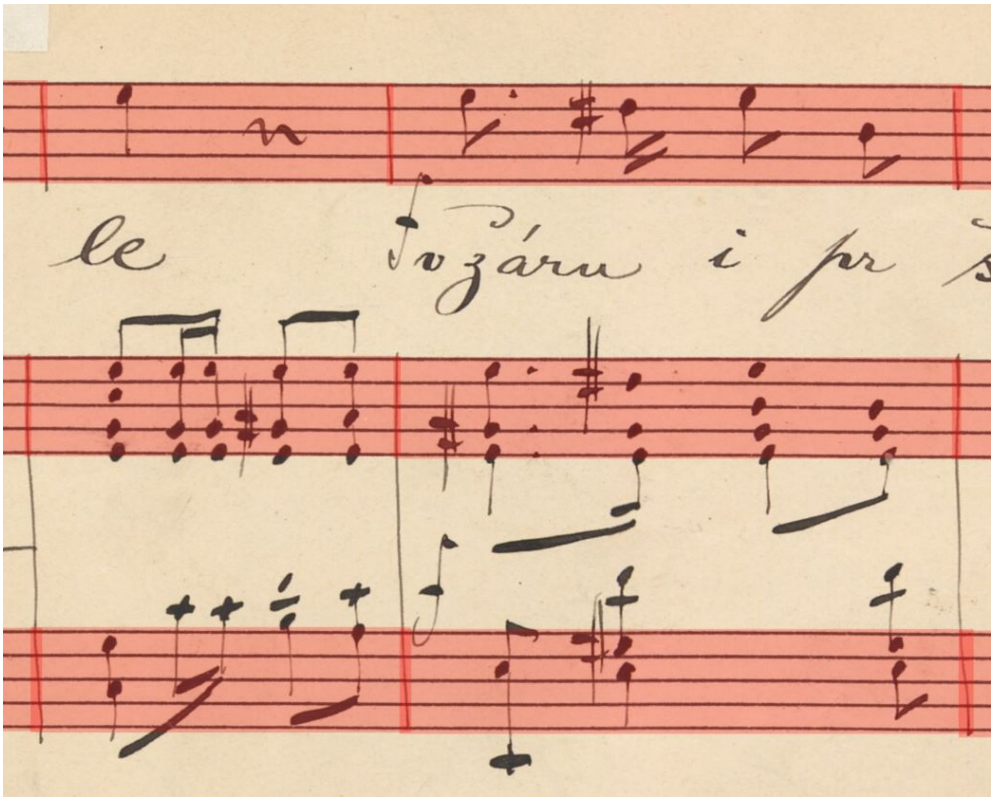
V kontextu syntézy notopisových dat se tomuto kroku říká postprocessing. Navržené metody jsou popsány v kapitole 3. Na fakultě se vyvíjí syntezátor ručně psaných not Mashcima [4] a tyto postprocessingové metody navrhujeme, vyvíjíme a testujeme za účelem jejich integrace do tohoto nástroje. K ověření pozitivního vlivu navržených metod na výsledný OMR model jsme navrhli sadu experimentů popsaných v kapitole Experimenty 5. Vyhodnocení provádíme na zjednodušené zástupné úloze "object detection", tedy detekci (rozpoznávání) objektů v obraze. V našem případě jsme se zaměřili na tři základní objekty, které však pokrývají celou škálu rozměrů různých hudebních symbolů. Těmito objekty jsou notové hlavičky, takty a osnovy, jak lze vidět na Obrázcích 1.2, 1.3, 1.4 a 1.5.



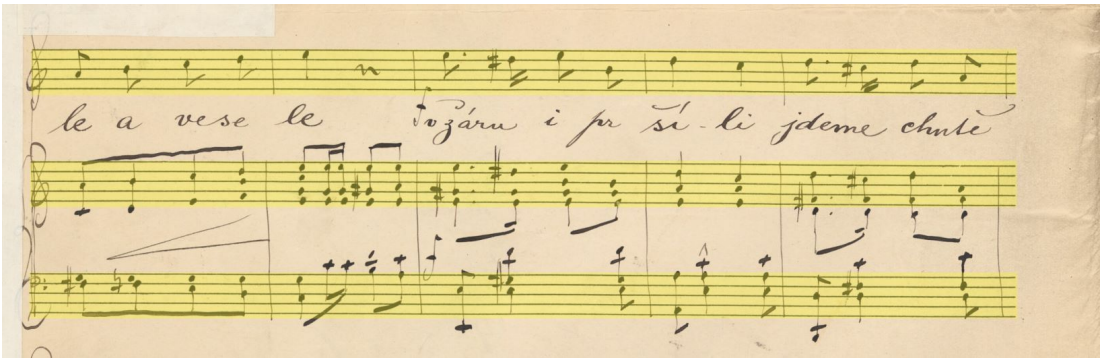
Obrázek 1.2 Anotace (vymezení bounding boxů) zkoumaných hudebních symbolů na skenu ručně psaného notopisu.



Obrázek 1.3 Anotace notových hlaviček.



Obrázek 1.4 Anotace taktů.



Obrázek 1.5 Anotace osnov.

Z provedených experimentů vyplývá, že všechny navržené metody mají pozitivní vliv na finální OMR model, nicméně největší přínos má generování syntetického pozadí notopisů. Z časových a technických důvodů se bohužel nepodařilo provést všechny plánované experimenty. Kvůli malé velikosti notových hlaviček se použitý model YOLOv8 4.4 nedokázal naučit rozpoznávat tento typ hudebního symbolu. Experimenty proběhly pouze na notách tištěných, protože zmiňovaný syntezátor Mashcima [4] ještě neumí produkovat bounding boxy objektů, a není tedy možné získat prvotní umělá anotovaná data, na kterých by byly aplikovány metody postprocessingu. Nicméně i přes veškeré nesnáze je z dostupných výsledků patrné, že navržené metody postprocessingu zlepšují schopnost učení modelů a je tak vhodné tyto metody přidat do současného syntezátoru Mashcima.

2 Související práce

2.1 Optical Music Recognition

Optical Music Recognition (OMR) je rozvíjející se oblast na rozhraní informatiky, hudební teorie a zpracování dokumentů [1]. Jejím cílem je umožnit automatické čtení hudebních notopisů. I přes svůj velký potenciál OMR zůstává velmi obtížnou oblastí výzkumu. To je z velké části dáno složitostí hudební notace a interdisciplinárními znalostmi, které jsou pro řešení této úlohy nezbytnou součástí znalostí výzkumníků. Calvo-Zaragoza, Hajič a Pacha ve své práci "Understanding Optical Music Recognition" [1] zavádějí rozsáhlou definici oblasti OMR, identifikují stěžejní výzvy a vytvářejí tím tak základní rámec pro budoucí pokroky.

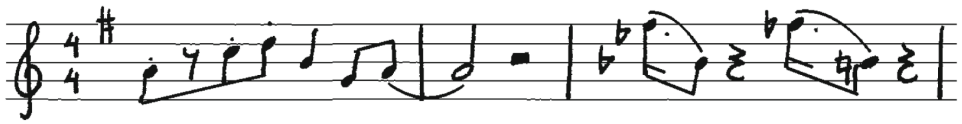
Kvůli spoustě existujících odlišných přístupů k oblasti OMR se velmi důležitým a nezbytným bodem jejich práce stává samotné sjednocení definic OMR. Calvo-Zaragoza a kol. navrhují definici, která v sobě zahrnuje cíl OMR, tj. umožnit počítačům výpočetně interpretovat hudební notaci z dokumentů. To zahrnuje nejen rozpoznávání grafických symbolů hudebních notací, ale také porozumění hudební sémantice. Tato definice slouží jako základ pro následné diskuse v celém jejich článku.

Calvo-Zaragoza a kol. se ve své práci [1] zmiňují o důležitosti integrace moderních technologií do oblasti OMR. Zejména pokud jde o metody hlubokého učení, které jsou prezentovány jako transformační faktor pro celou oblast OMR, mající potenciál výrazně zlepšit přesnost a efektivitu OMR systémů. Naše práce zabývající se postprocessingem syntetických trénovacích dat, která jsou kriticky důležitá pro úspěšné trénování hlubokých neuronových sítí, tak navazuje na myšlenky a definice práce prezentované Calvo-Zaragozou a kol. [1].

2.2 Mashcima

Jak již bylo zmíněno v Úvodní kapitole 1, tato práce má za cíl vylepšit syntézu umělých dat pro problém Optical Music Recognition (OMR). Prvním konkrétním cílem je našimi metodami postprocessingu rozšířit právě vyvíjený nástroj fakulty. Jedná se o syntezátor notopisů Mashcima jehož autory jsou Mayer a Pecina. V svém článku z roku 2021 [4] popisují řešení problému OMR, a to konkrétněji v oblasti ručně psaných not, tedy HMR (Handwritten Music Recognition). Kvůli velmi vysokým nákladům na ruční anotaci a s tím spojenému silnému nedostatku tréninkových dat, vyvinuli Mashcimu, jako generátor realistických obrázků ručně psané hudby, které lze použít k tréninku modelů strojového učení pro HMR.

Systém Mashcima [4] využívá anotací symbolů z datasetu MUSCIMA++ [5]. I díky tomu umožňuje generování obrázků s odlišnými styly rukopisů různých skladatelů, což je klíčové pro rozmanitost, a tím pádem kvalitu tréninkových dat. Pro usnadnění samotného generování notopisů bylo v rámci vyvíjení Mashcimy definováno nové kódování *Mashcima encoding*. To je založené na již dříve existujících formátech kódování, ovšem přináší zjednodušení tagů, zdokonalení reprezentace výšky tónů a přidání značek pro hudební dynamiku a artikulace (staccato, legato aj.) [4]. Na Obrázku 2.1 je ukázka vygenerovaného obrázku systémem Mashcima s příslušným kódováním Mashcima encoding daného obrázku.



```
clef.G-2 time.4 time.4 #4 e=-1 . er =e=1 . =e2 . q0 e=-2 =e-1 ( |
   ) h-1 hr | b0 b3 s=4 * ( ) =e0 qr b4 s=4 * ( ) N0 =e0 qr |
```

Obrázek 2.1 Uměle vytvořený notopis systémem Mashcima [4] napodobující ručně psané noty s kódováním daného notopisu.

Samotné generování syntetických dat sestává z několika fází rozvržených takovým způsobem, aby mohlo docházet k vytváření realistických tréninkových datasetů. Kroky jsou následovné [4]:

- Zisk masek hudebních symbolů (Acquisition of music symbol masks), které probíhá jejich extrahováním z datasetu MUSCIMA++ [5]. Tyto masky reprezentují jednotlivé prvky hudební notace, se kterými lze manipulovat a znovu je použít v různých konfiguracích.
- Zisk anotací symbolů (Obtaining ground-truth annotations), zahrnující vznik jejich kódování ve zmíněném Mashcima encoding.
- Rozmístění symbolů na osnovu (Symbol placement on a staff) takovým způsobem, aby replikovalo nějakou přirozenou variabilitu vyskytující se v autentických ručně psaných notopisech. Dochází tedy k práci s rozestupy, zarovnáním nebo interakcí jednotlivých symbolů.
- Vykreslení obrázku (Image rendering), které zahrnuje například i jemné úpravy tloušťky čar symbolů, případně rozmazání tak, aby obrázek co nejvíce připomínal ručně psané noty.

Tento přístup umožňuje autorům generovat velké objemy rozmanitých a realistických syntetických dat pro trénink systémů HMR, které jsou při trénování modelů velice úspěšné [4]. Mashcima je však zatím schopna generovat jen monofonní notopisy, neboli notopisy takové, které mají jen jednu melodickou linku. Stávající forma Mashcimy generuje obrázky pouze černo-bílé, a není tak schopna napodobit autentické ručně psané notopisy v plné šíři.

Integrací metod této práce do systému Mashcima tak dáme možnost vzniku syntezátoru napodobujícího daleko širší škálu rukopisných notopisů. Tento rozšířený syntezátor pak bude schopen tvořit trénovací datasety pro modely hlubokého učení, které budou schopny úspěšně řešit jak problémy OMR, tak konkrétnější úlohu HMR.

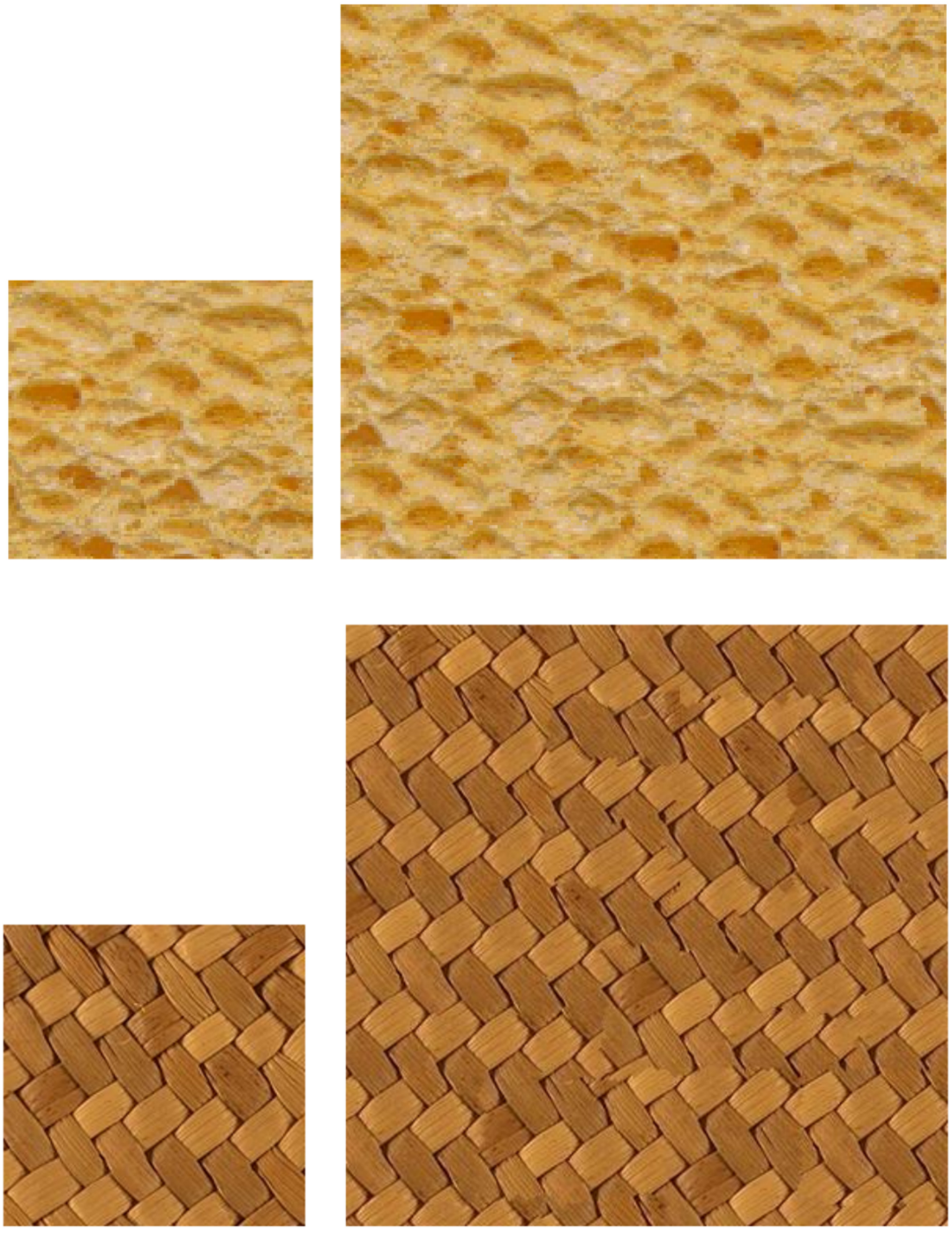
2.3 Image Quilting for Texture Synthesis

Článek Efrose a Freemana [6] představuje novou techniku generování obrázků nazvanou *image quilting*. Tato metoda syntetizuje nové obrázky spojováním malých částí z již existujících obrázků. Obsah jejich práce má dvě hlavní témata, kterými jsou syntéza textury a přenos textury. Technika přenosu textury spočívá v mapování textury z jednoho objektu na jiný objekt na základě korespondenční mapy. Ta zahrnuje shodu lokálních vlastností (například intenzita) mezi zdrojovým a cílovým obrázkem, aby byla zajištěna bezproblémová integrace textur. Pro tuto práci jsme však využili zejména část článku zabývající se syntézou textur, jejíž hlavní složkou je image quilting.

Algoritmus *Image Quilting* prezentovaný Efrosem a Freemanem [6] představuje významný pokrok v oblasti syntézy textur. Základním konceptem tohoto algoritmu je syntéza nových textur spojováním malých částí z existujícího obrázku, ovšem bez viditelných spojů. Vstupní obrázek textury je rozložen na více překrývajících se čtvercových bloků (dále jen čtverců). Nový obrázek textury je pak vytvořen umísťováním těchto čtverců vedle sebe. Aby se zabránilo viditelným "švům", čtverce se nepatrně překrývají. A aby se zajistila konzistence napříč překryvy hran čtverců, jednotlivé postupně přidávané čtverce nejsou vybírány náhodně. Výběr probíhá takovým způsobem, aby každý následující čtverec minimalizoval chybu v oblastech překryvu. Tento krok zahrnuje detailní výpočet, který kvantifikuje míru odlišnosti mezi hranami čtverců pomocí kvadratického rozdílu mezi hodnotami pixelů přes daný překryv.

Samotná syntéza textury má za cíl vygenerovat větší texturu z malého vzorku tak, aby byla zachována vizuální charakteristika původního vzorku. Efros a Freeman ve svém článku [6] zdůraznili rozdíl mezi úspěchem syntetizovaní textur stochastických a strukturovaných, jejichž rozdíl lze vidět na Obrázku 2.2. U strukturovaných textur, tedy u takových textur, které mají nějaký jasný opakující se vzor, je daleko více znát případná nedokonalost ve vzniklých "švech"mezi jednotlivými čtverci. Naše práce však využívá syntézy převážně stochastických textur, které jsou daleko více náhodné a bez nějakého konkrétního vzoru. Právě pro tyto stochastické textury algoritmus Image Quilting funguje nejlépe, neboť náhodnost textury skvěle koresponduje s náhodným vybíráním a umísťováním čtverců.

Další výhodou algoritmu Image Quilting [6] je rychlost a efektivita jeho výpočtů oproti starším technikám syntetizování textur, uvedeným v předchozí práci Efrose a Leunga [7]. To vše neprobíhá na úkor kvality syntetizovaných obrázků, ale právě naopak. Pro tuto práci je tak Image Quilting ideálním přístupem k implementaci degradace pozadí, jež je popsána v další kapitole 3.2.



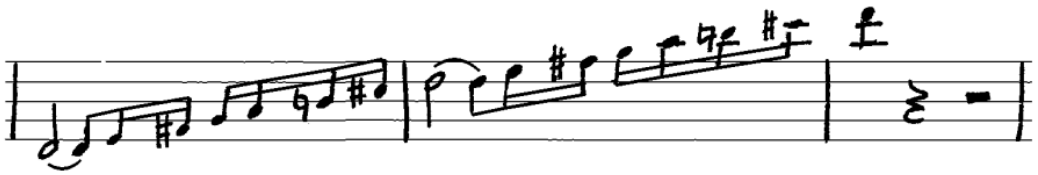
Obrázek 2.2 Syntéza stochastické (nahoře) textury (vpravo) podle vzorku textury (vlevo) a syntéza strukturované (dole) textury (vpravo) podle vzorku textury (vlevo) pomocí algoritmu Image Quilting [6].

3 Postprocessing

Účelem postprocessingu je vyřešit problém s nedostatkem trénovacích dat a zároveň zlepšit jejich kvalitu, a to konkrétně pro úlohu optického rozpoznávání notopisů neboli Optical Music Recognition (OMR). Postprocessing je poslední fází tvorby syntetických obrázků notopisů, která má za cíl převést černo-bílý obrázek na barevný tak, aby simulovala vzhled skenu fyzického dokumentu.

Obrázky, se kterými postprocessing pracuje, jsou striktně černo-bílé. Jedná se o různé symboly v černé barvě, které se vyskytují na bílém pozadí. Cílem postprocessingu je z těchto černo-bílých obrázků vytvořit nejen obrázky barevné, ale hlavně obrázky takové, aby se co nejvíce přibližovaly realistickému vzhledu. V případě notopisů je realistickým vzhledem myšleno co nejvěrnější napodobení originálních hudebních záznamů, často ručně psaných.

Pokud srovnáme uměle vytvořený syntetický notopis vygenerovaný počítačem na Obrázku 3.1 se skenem ručně psaného autentického notopisu na Obrázku 1.1, je znát skutečně velký rozdíl. Na první pohled si člověk u syntetického notopisu všimne, že zde schází jakékoliv pozadí, které by imitovalo přirozenou strukturu papíru. Ke struktuře papíru patří i například jeho tenkost a s tím spojená průsvitnost. Dalším silným jevem na reálném notopisu je tak třeba viditelný prosak zadní strany, který může být někdy tak silný, že by se dal snadno splést s validními informacemi na zkoumané stránce.



Obrázek 3.1 Příklad syntetického notopisu, který je vstupem pro úlohu postprocessingu. Notopis je černo-bílý a obsahuje hudební informaci, ale vzhledem se ještě nepodobá skenu fyzického dokumentu.

Kromě problémů viditelných z většího měřítka jsou při detekci objektů podstatnou složkou i drobnosti, které by mohly zapříčinit zmatení v rámci rozpoznávání symbolu. Může se jednat o inkoustové kaňky, porušenou strukturu papíru, odlišné rukopisy různých autorů nebo nekvalitní pořízení fotografie či skenu s následným šumem na obrázku. Tyto jevy vyskytující se na reálných rukopisech nazveme *degradacemi*.

K úspěšnému výsledku úlohy postprocessingu je třeba tyto degradace definovat a pokusit se je co nejlépe napodobit.

3.1 Degradace

Jak je popsáno výše, pojem degradace obrázku v kontextu postprocessingu znamená jakákoliv odchylka od výchozího černo-bílého obrázku s informacemi v barvě černé na pozadí bílé barvy. Jednotlivé degradace byly definovány takovým způsobem, aby se nejednalo o ojedinělé jednotky případů, ale aby se daný jev vyskytoval na autentických notopisech často. Hlavním zdrojem reálných ručně psaných notopisů byl digitální archiv Moravské zemské knihovny [8].

3.2 Textura papíru pozadí

Ručně psané notopisy pocházejí z různých dob, jsou psány různými autory, a tak jsou pochopitelně psány i na různé druhy papírů. Na první pohled jsme se nedomnívali, že by byly rozdíly tak markantní, ale po bližším zkoumání řady skenů a fotografií rukopisných notopisů jsme zjistili, že jsou odlišnosti skutečně významné. Prvním zásadním rozdílem je barva papíru, která se pohybuje v rozmezí různých odstínů nejběžnější béžové/hnědé/nažloutlé, ovšem může být třeba i v odstínech modré, červené i dalších už méně častých výjimek. Tyto extrémní rozdíly barev mohou být i následkem rozdílného způsobu snímání skenů nebo fotografování, kdy hraje roli i špatné nasvícení notopisů aj.

Dalším rozdílným prvkem různých pozadí, který je prakticky nezávislý na barvě, je samotná textura pozadí. Často není pozadí tvořeno jednolitou barvou a homogenní strukturou, ale obsahuje různá drobná poškození, tmavější nebo světlejší místa nebo zbytky různých materiálů, ze kterých je papír tvořen. K těmto sekundárním jevům dochází zpravidla i běžným používáním a opotřebováním materiálu papírů. Papír tak může být v některých místech třeba i tenčí, a tím pádem průsvitnější. Všechny tyto rozličnosti zahrnuje degradace s názvem textura papíru pozadí.

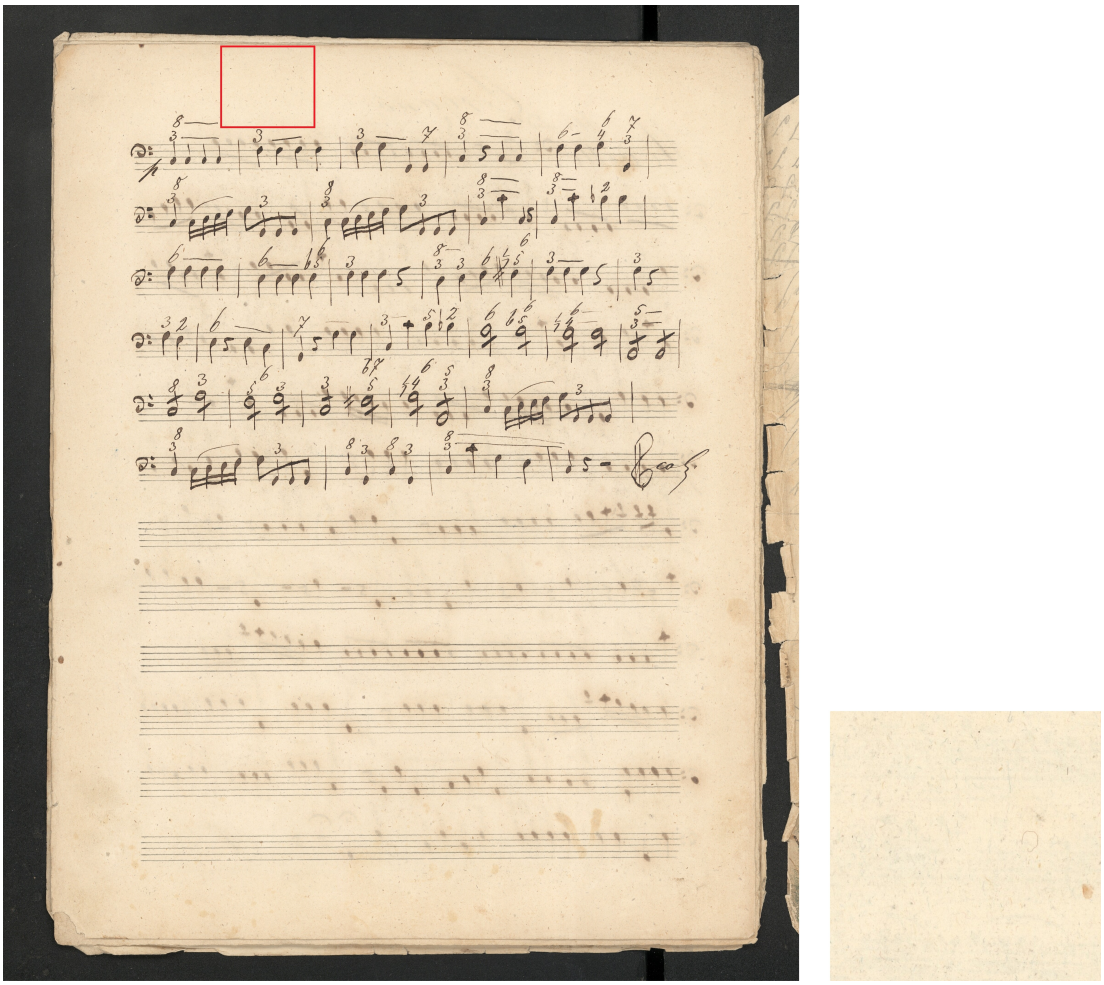
Implementace

Vytváření textur papíru pozadí bylo inspirováno prací Efrose a Freemana [6]. Ve svém článku popisují tzv. *image quilting*, což v zásadě znamená vytvoření nového obrázku vhodným seskládáním jednoho malého vzorku již existujícího obrázku. Na Obrázku 3.2 můžete vidět výběr vzorku textury ze skenu autentického rukopisu. Na Obrázku 3.3 je pak výsledek image quiltingu.

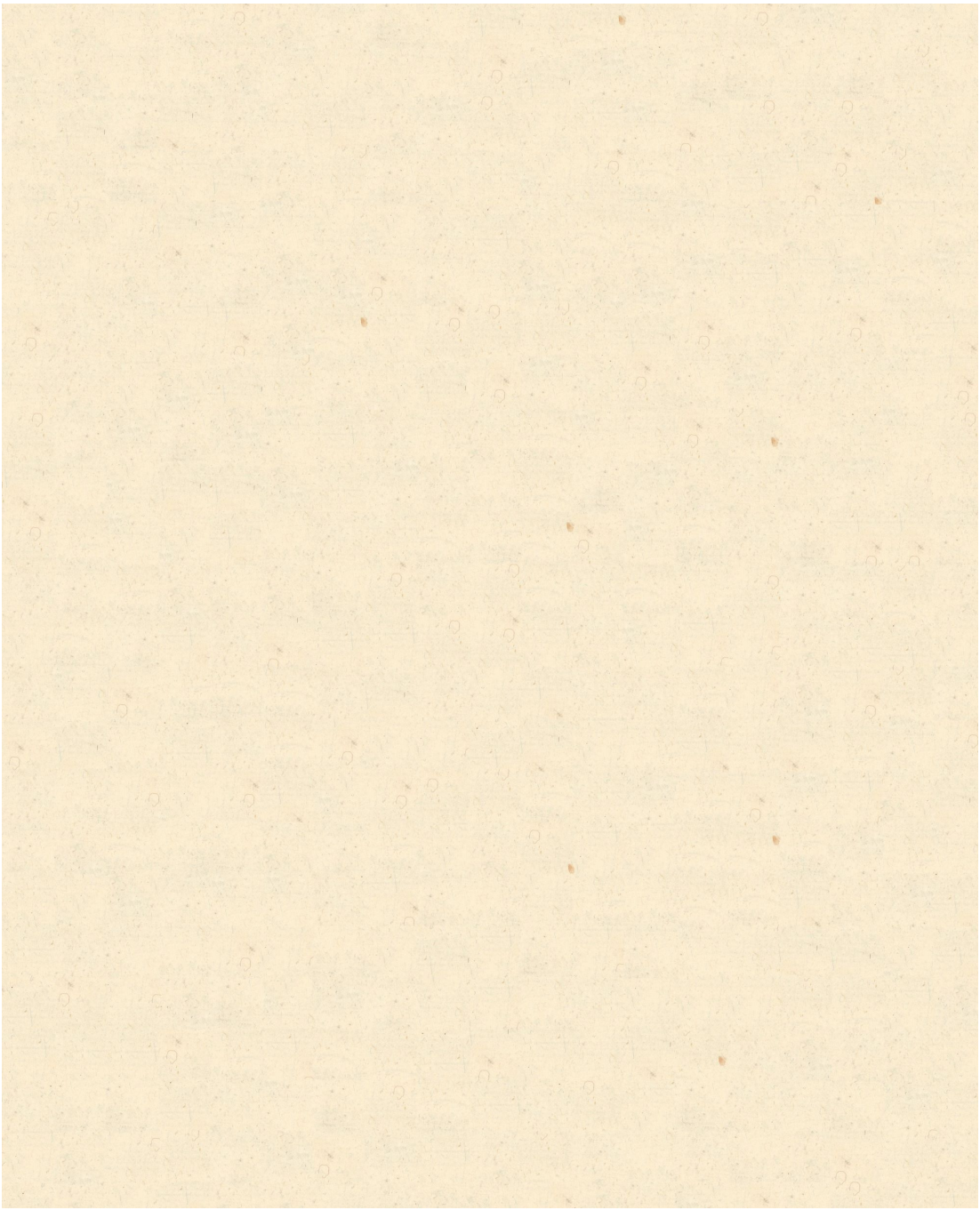
Image quilting spočívá v tom, že ze vstupního obrázku, na kterém je zachycena požadovaná textura, chceme získat další obrázky, často několikanásobně větších rozměrů [6]. Nejjednodušším způsobem by bylo vytvořit ze vzorku několik náhodných menších obrázků a ty pak umisťovat vedle sebe, dokud nedosáhneme požadovaného rozměru pozadí. Důležitou podmínkou je však to, aby získané obrázky vypadaly realisticky. Zde nastává problém, neboť po pouhém nalepení vzorku textury vedle sebe máme velmi zřetelné hranice vzorků tak, jako je vidět na Obrázku 3.4.

Aby mohl být image quilting [6] úspěšný, musíme zabránit viditelnosti hranic mezi jednotlivými vzorky naskládanými vedle sebe. Tento problém se dá vyřešit tak, že vzorky nebudeme pokládat vedle sebe náhodně, ale z připravených menších obrázků ze vzorku vybereme ten jeden nejvhodnější, který se hodí ke svému sousedovi nejlépe. Při výběru následujícího pokládaného obrázku chceme minimalizovat rozdílnost společné hrany. To zajišťujeme kvantifikováním chyby, jinými slovy také míry odlišnosti hran jednotlivých čtverců, viz 2.3. Následně vybereme takový obrázek, který má tuto chybu vzhledem k již položenému obrázku nejmenší.

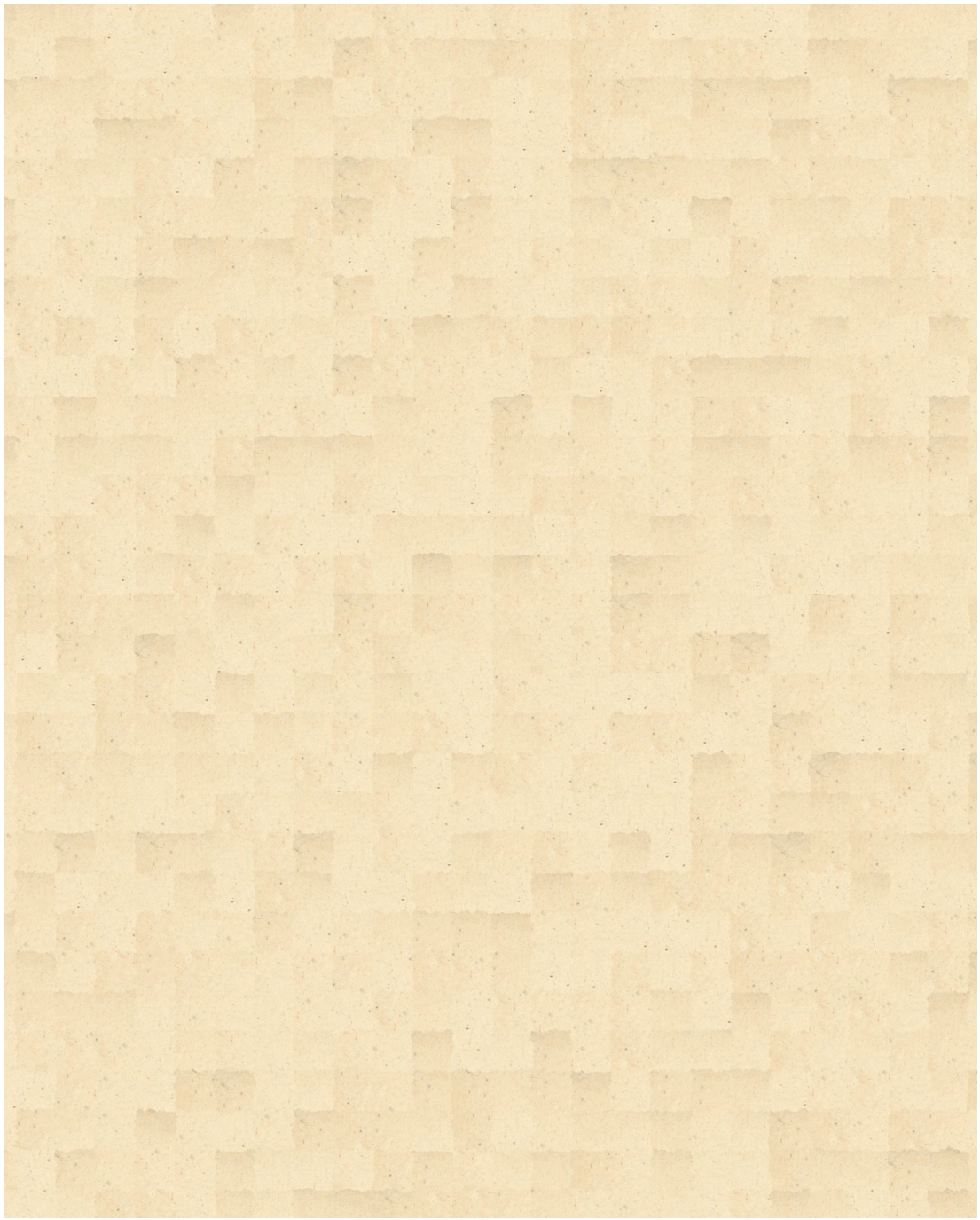
Zmiňovaný krok kvantifikace chyby není v našem případě třeba optimalizovat. To jest z důvodu využívání čistě stochastických typů textur, jako je popsáno v kapitole 2.3. Algoritmus běží s optimalizací výběru čtverců řádově v jednotkách sekund, někdy až 10 sekund. To je v praxi bohužel příliš pomalé, a tím pádem nepoužitelné. Bez zmiňované optimalizace, která je nutná jen pro strukturované textury, se dostáváme k době běhu algoritmu řádově ve stovkách milisekund. Zrychlení je tak až stonásobné. V námi generovaných typech textur optimalizace nutná není, a tak dáme přednost šetření času.



Obrázek 3.2 Výběr vzorku textury (vlevo). Vybraný vzorek textury pozadí (vpravo).



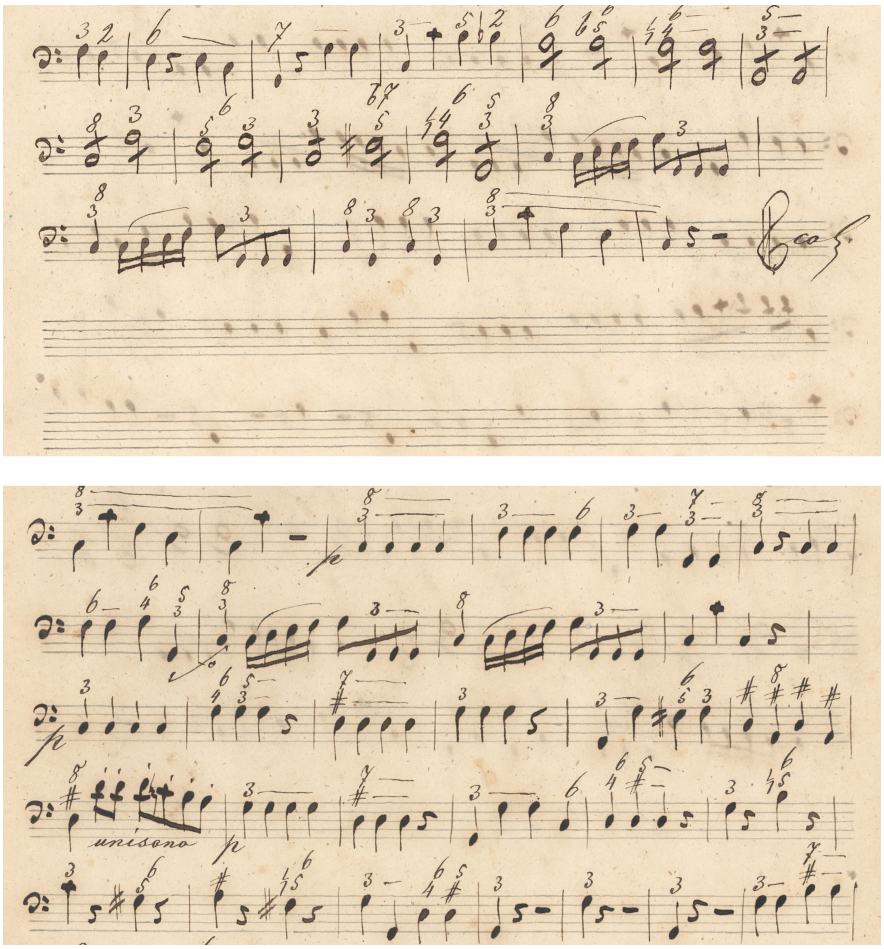
Obrázek 3.3 Výsledná textura z image quiltingu.



Obrázek 3.4 Jednoduché použití vzorku.

3.3 Prosak zadní strany

Velmi významnou degradací je rovněž prosak zadní strany. Jedná se o jev, kdy vlivem příliš silné vrstvy inkoustu nebo naopak velmi jemného papíru dojde k prosvítání symbolů na druhé straně papíru. To je problém hlavně z toho důvodu, že při rozpoznávání objektů na stránce může snadno dojít ke zmýlení a momentálně nezajímavá informace z opačné strany se vyhodnotí jako součást právě zkoumané stránky. Protože historicky se notopis psal vždy oboustranně, může se jednat o velmi častý problém při rozpoznávání symbolů na stránce. Na Obrázku 3.5 si lze všimnout reálného notopisu s viditelným prosakem zadní strany papíru. Tento prosak můžeme porovnat s obsahem druhé strany papíru na tomtéž Obrázku 3.5, kde jsou ony prosáknuté symboly.



Obrázek 3.5 Reálný notopis s prosáknutím opačné strany papíru (nahoře). Opačná strana papíru notopisu (dole).

Implementace

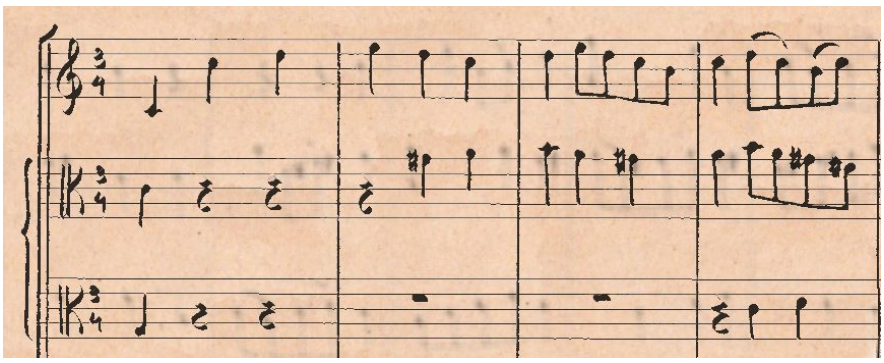
Tento typ degradace je pro implementaci relativně snadný. Důležité je si uvědomit, co vlastně druhá strana papíru obsahuje. Nejčastěji se tam bude nacházet rovněž notopis. Může se jednat i o pouhé texty či úvodní strany vydání, ale tento jev příliš častý není. Budeme tedy předpokládat, že na druhé straně papíru bude notopis.

Jako vstupní obrázek tedy máme černo-bílý notopis, případně s již doplněným pozadím z předchozího bodu 3.2, na tom nezáleží. Jak jsme si uvedli, chceme simulovat prosak druhé strany, kde se bude nacházet také notopis. Vybereme si tedy další jiný obrázek, který budeme chtít umístit na opačnou stranu papíru a následně ho prosáknout. Nejčastější způsob psaní na notový papír je na výšku tak, aby se dalo svázanými stranami listovat. Tedy pokud budou noty skutečně na výšku, chceme prosakovaný obraz překlopit podél dlouhé strany. Pokud by se jednalo o notopis, který je na šířku, stále platí pravidlo, že při listování stranami nechceme otáčet celým svazkem. V tomto případě by šlo tedy o překlopení podél krátké strany. Obě možné situace lze však vyřešit jedním případem, a to tak, že do středu vybraného prosakovaného obrázku umístíme svislou osu. Následně provedeme symetrické otočení obrazu podle této osy.

Nyní potřebujeme spojit výchozí obrázek jako přední stranu papíru s připraveným obrázkem jako prosak zadní strany. Oba obrázky mají pozadí, ať už je to čistě bílá, nebo již připravené pozadí ve formě textury z předchozího bodu degradací 3.2. Vzhledem k tomu, že nechceme jeden obrázek tím druhým úplně překrýt, ale pouze přidat další symboly v černé barvě, musíme přidávaný prosak zbavit pozadí. Tedy bílou barvu nahradíme průhlednou, přidáme tak alfa-kanál. Nyní už stačí jen zajistit, aby přidávané symboly vypadaly jako prosáknuté ze zadní strany.

K vytvoření dojmu prosáknutí potřebujeme nezbytně dvě úpravy připravených černých symbolů na obrázku s průhledným pozadím. Symboly, tedy celý obrázek, musíme lehce rozmazat. Prosak totiž jen velmi zřídka zachovává ostré hrany a rohy. Druhým nezbytným bodem je úprava syté černé barvy symbolů. Papír má určitou tloušťku, a tak při prosánkutí barvy část tekutiny pohltí a nepropustí ji. Výsledná barva je pak o několik odstínů světlejší než barva původní. Napodobení tohoto jevu tak nejlépe dosáhneme, pokud černé symboly lehce zprůhledníme. Intenzita zprůhlednění se může pohybovat v různých mezích, naším měřítkem je co nejrealističtější dojem z celého výsledného notopisu.

Podobně jako u první zmiňované degradace 3.2, kde můžeme nasyntetizovat libovolné množství rozdílných typů textur pomocí výběru různých vzorků, i zde u prosaku máme velkou variabilitu v jeho provedení. Jednak může být prosakovaný obrázek pokaždé jiný a jednak se intenzita prosaku, tedy poměr černé a průhledné barvy prosakovaných symbolů, může s každým obrázkem měnit. Na Obrázku 3.6 je možný výsledek původně černo-bílého obrázku po dodání prosaku a vytvořené textury na pozadí.



Obrázek 3.6 Syntetický černo-bílý notopis s uměle dodaným prosakem a vytvořenou texturou na pozadí.

3.4 Kaligrafický rukopis

Jako další typ degradace černo-bílého syntetického notopisu, která slouží k co nejlepšímu přiblížení se reálným ručně psaným autentickým notopisům, si představíme napodobování rukopisu. Styly rukopisů jsou různé a hlavním rozdílem ve výsledných notopisech s odlišnými autory jsou nepatrné odchylky tvarů psaných symbolů. Tento problém odchylky tvarů však již většinou pokrývá samotné vygenerování prvotního černo-bílého obrázku. Proto se zaměříme na jinou vlastnost rozdílných rukopisů, a to sklon pera s intenzitou tlaku na papír při psaní. Můžeme si to představit jako typické kaligrafické písmo, kde je v jedné části symbolu čára širší a v jiné výrazně tenčí, jako je vidět na srovnání v Obrázku 3.7.

Obrázek 3.7 Ukázka různého sklonu a tloušťky pera na symbolu houslového klíče.

Implementace

Hlavní složkou napodobení vzhledu ručně psaného textu jsou dvě známé transformace - *dilatace* a *eroze*. Oba procesy mají své transformační jádro, kterým je specifická matice. Dilatační jádro vyobrazené ve výrazu 3.1 má za následek rozšíření černých symbolů na obrázku v definovaném směru, představujících inkoust. Jeho výchozí struktura je definována jako svislá čára, ale v průběhu algoritmu může dojít k rotaci této čáry kolem svého středu za účelem implementace různých sklonů písma.

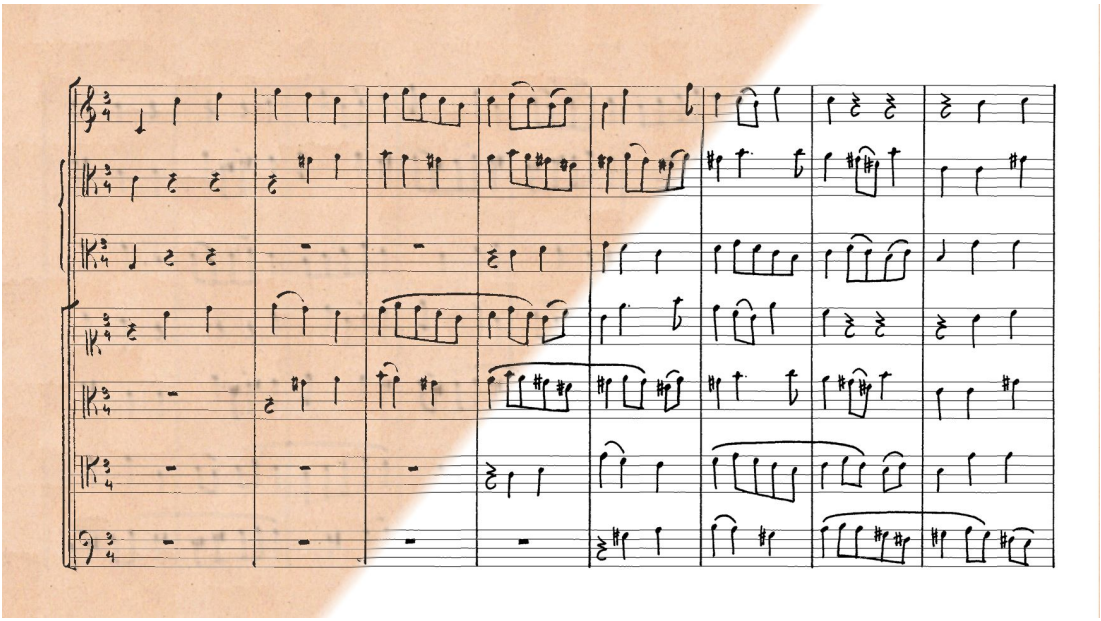
$$\begin{pmatrix} \mathbf{0}, \ldots, 0, 1, 0, \ldots, 0 \\ \vdots \\ \mathbf{0}, \ldots, 0, 1, 0, \ldots, 0 \end{pmatrix} \tag{3.1}$$

Eroze oproti tomu funguje opačně, tedy zmenšuje požadované oblasti ve směru definovaném v erozním jádru uvedeném ve výrazu 3.2. Výchozí struktura jádra pro erozi je definována opačně než pro dilataci, tedy jako vodorovná čára. Stejně jako u dilatace může být směr čáry, a tedy směr eroze, změněn pomocí rotace kolem středu. Oba procesy dohromady dodávají obrázku realistické vzezření, které odráží nepravidelnost a plynulost lidského rukopisu.

$$\begin{pmatrix} \mathbf{0}, \ldots, 0 \\ \vdots \\ \mathbf{0}, \ldots, 0 \\ \mathbf{1}, \ldots, 1 \\ \mathbf{0}, \ldots, 0 \\ \vdots \\ \mathbf{0}, \ldots, 0 \end{pmatrix} \tag{3.2}$$

Jako výchozí obrázek máme opět černo-bílý syntetický notopis. Pro provedení výše zmíněných operací dilatace a eroze je nutné obrázek nejprve invertovat tak, abychom měli černé pozadí a transformace prováděli na bílé barvě, která momentálně představuje inkoust. Jako další možný krok lze provést rotaci připravených jader, jejíž míra zajišťuje odlišný finální sklon písma. Tedy podobně jako u dříve popsaných degradací je i zde možnost parametrizace tak, aby po aplikaci degradace rukopisu nevypadal každý notopis v tomto směru stejně. Následně s připraveným dilatačním a erozním jádrem tak může obrázek projít samotnými procesy dilatace a eroze. Zmíněné dilatační jádro rozšíří invertované bílé oblasti v požadovaném směru definovaném v jádře. Následná eroze upraví bílé expandované oblasti v požadovaném směru a ztenčí je do tvarů připomínající tahy štětcem či pera.

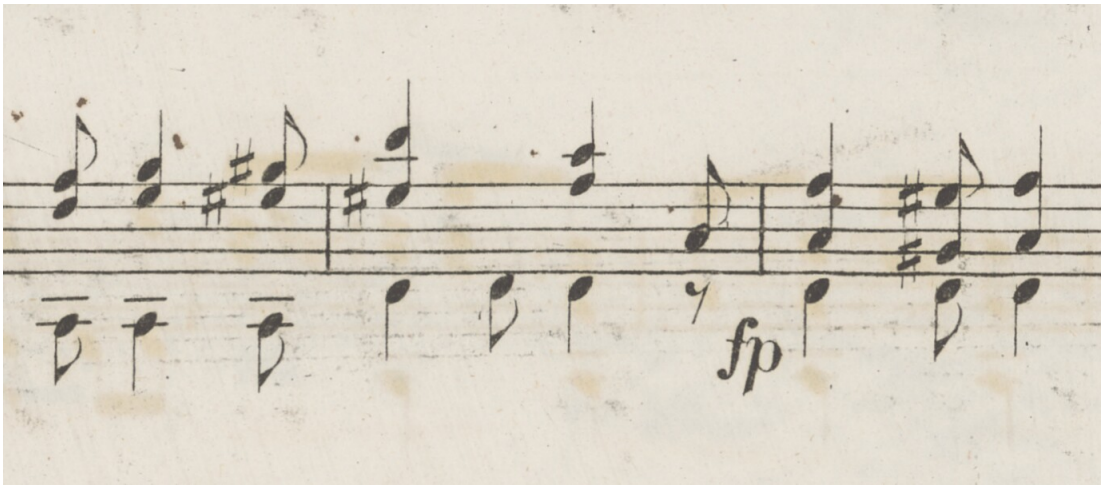
Posledním krokem už je jen zpětné invertování obrázku, po kterém budou symboly v barvě černé a pozadí v barvě bílé. Výsledný transformovaný obrázek má místo stejných rovnoměrných čar unikátní tahy perem se specifickým sklonem psací potřeby napodobující různé rukopisy díky parametrizaci směru dilatace a eroze. Nejlépe viditelné degradace rukopisu si lze povšimnout třeba u čtvrťových pomlk, klíčů, ligatur nebo obecně jakéhokoliv symbolu, který se skládá z více linek různých směrů. Po přidání dalších degradací jako třeba textury pozadí a prosaku opačné strany, jako je na Obrázku 3.8, se už začínáme blížit realističnosti napodobenin pravých notopisů.



Obrázek 3.8 Srovnání výchozího černo-bílého obrázku s degradovaným pomocí přidání textury, prosaku a rukopisu.

3.5 Kanungo Šum

Jednou z posledních nezbytných vlastností, která schází syntetickým rukopisům k co nejvěrnějšímu napodobení reálných notopisů, je zachycení určitého stáří notových záznamů. Pokud se bavíme o čistě rukopisných notopisech bez použití knihtisku, jejich vznik se může datovat třeba až do 15. století n. l. Zmiňované stáří se tak podepisuje hlavně ve formě postupně blednoucího či na některých místech úplně chybějícího inkoustu. Kromě mizejícího inkoustu se vlivem používání notopisů objevují na papíru nepatrné poruchy, nahromaděný prach a další nečistoty, které postupem času vytvářejí malé tečky. Tyto tečky navíc společně například se špatným snímáním obrázku vytvářejí takzvaný šum, jak můžete vidět na Obrázku 3.9.



Obrázek 3.9 Zašumělý obrázek, kde šumem se rozumí bílé flíčky na černých symbolech a černé flíčky na světlém papíru. Tento jev je simulován pomocí Kanungo šumu [9].

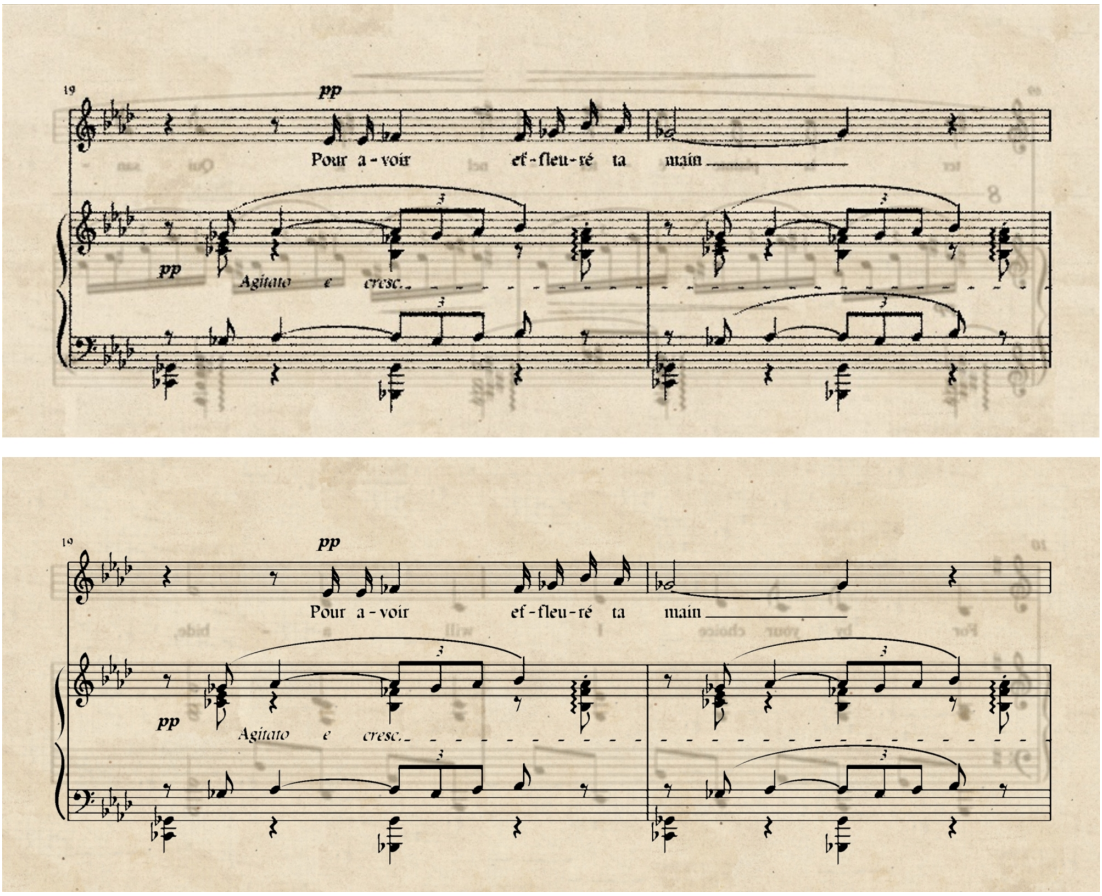
Implementace

My se budeme snažit tento šum napodobit. Typů šumů je spousta, my budeme používat *Kanungo* šum z článku Kanunga a kol. [9]. Kanungo šum je pojmenován po Tapasi Kanungovi, který je autorem tohoto algoritmu. Vyvinul jej pro simulaci degradace dokumentů kontrolovaným způsobem. Jeho účelem je tak napodobit skutečné opotřebení, jako jsou šmouhy, roztírání inkoustu a zrnitost papíru, které staré dokumenty obvykle vykazují. Pro náš problém napodobení stáří notopisů tak bude velmi vhodný.

Jako vždy máme na počátku aplikace libovolné degradace černo-bílý obrázek jako vstup algoritmu. Přestože je obrázek již černo-bílý, bude považován, že je barevný (ve formátu RGB - Red, Green, Blue), neboť právě RGB je výchozí formát obrázků. Proto proces začíná převedením černo-bílého obrázku na obrázek ve stupních šedi. Toto zjednodušení reprezentace obrázku snižuje složitost, přičemž se algoritmus zaměřuje na texturu a kvalitu obrázku spíše než na jeho barvu. Aplikací prahu je obrázek v stupních šedi dále zjednodušen na binární formát černo-bílý, tedy jasným rozdělením, které odstíny šedé budou definovány jako bílá a které odstíny šedé budou definovány jako černá.

Algoritmus provádí transformaci vzdálenosti jak na popředí (prvky), tak na pozadí [9]. Tato transformace poskytuje mapu nejkratších vzdáleností od jakéhokoli bodu k nejbližší hranici mezi popředím a pozadím (černou a bílou), což je klíčové pro určení, jak by měla degradace po celém obrázku kolísat. V zásadě čím blíže jsme nějakému objektu, tedy symbolu, tím více šumu bychom chtěli vytvořit. S využitím map vzdáleností algoritmus vypočítá pravděpodobnosti pro změnu (překlopení) každého pixelu z černé na bílou nebo naopak. Tyto pravděpodobnosti jsou modelovány pomocí exponenciální funkce, která zohledňuje vzdálenost od nejbližšího okraje [9]. Pixely blíže k okrajům mají vyšší pravděpodobnost překlopení, což simuluje degradaci související s okrajem, jako je roztírání inkoustu.

Nakonec musí pro nejlepší napodobení stárnutí papíru hrát roli při otáčení jednotlivých pixelů i náhoda. Vygeneruje se matice náhodných čísel, kde její prvky odpovídají pixelům obrázku. Pixely, jejichž odpovídající hodnoty v matici pravděpodobnosti přesahují práh spočítaný z obou vzdálenostních map, jsou překlopeny. Tím se změní původní obrázek tak, aby odrážel typické deformace stárnutí [9]. Posledním případným krokem je převedení obrázku v binární reprezentaci zpět do barevného formátu pro výstup. Obrázek sám o sobě však zůstává černo-bílý, ovšem oproti vstupnímu obrázku je nyní obohacen o efekty simulace stárnutí, viz Obrázek 3.10.



Obrázek 3.10 Aplikace šumu Kanungo [9] na degradovaný tištěný notopis. Srovnání notopisu s přidaným Kanungo šumem (nahoře) a notopisu bez přidání Kanungo šumu (dole).

4 Metodologie

Tato kapitola poskytuje ucelený přehled o metodologii práce zaměřené na zlepšení přesnosti a efektivity úlohy Optical Music Recognition (OMR) prostřednictvím postprocessingových metod popsaných v kapitole 3. A to včetně detailního popisu jednotlivých kroků a použitých technik. Hlavním cílem je vyhodnotit, zda vůbec a případně které postprocessingové techniky mohou napomoci zlepšení výsledků OMR. Kapitola se zabývá strukturou experimentů (sekce 4.1), výběrem a přípravou datasetů (sekce 4.2), použitým modelem YOLOv8 (sekce 4.4) a specifickými metodami hodnocení.

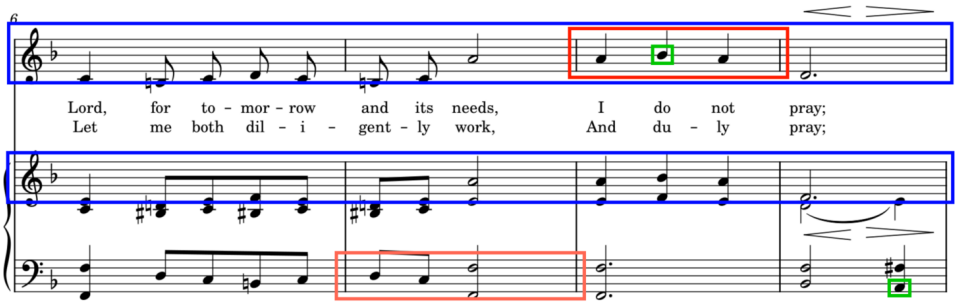
4.1 Struktura experimentů

Cílem naší práce je vymyslet a implementovat postprocessingové metody, které zlepší přesnost a efektivitu úlohy Optical Music Recognition (OMR). V rámci prováděných experimentů se tak zaměřujeme na vyhodnocení, jestli a které postprocessingové techniky mohou OMR pomoci. Současné OMR modely jsou velmi složité a jejich trénování může probíhat až v řádech jednotek dnů. Rozhodli jsme se tak vyhodnocovat postprocessing na jednodušší zástupné úloze, kterou je rozpoznávání objektů (object detection). Tato úloha spočívá v rozpoznání daných objektů v obrázcích. Jednotlivé detekované objekty jsou ohraničeny pomocí obdélníků, neboli tzv. *bounding boxes*, jak je ilustrováno na Obrázku 4.1. Tato úloha byla pro řešení úloh OMR používána již před nástupem end-to-end modelů, jak píše ve své práci Hajič [10]. Jak je zmíněno v práci Pachy [11], existují podúlohy OMR, jako například analýza obsahu stránky (rozpoznání osnov a taktů), kde i dnes zůstává úloha rozpoznávání objektů ideálním řešením.

Obrázek 4.1 Object detection, neboli rozpoznávání objektů [12]. Na obrázku je rozpoznán objekt člověk (vlevo) a dopravní kužely (vpravo), každý z objektů je zvýrazněn pomocí bounding boxu s mírou pravděpodobnosti, že se skutečně jedná o daný předpovězený objekt.

Pro naše experimenty jsme tedy vybrali úlohu detekce objektů. Konkrétně se zaměřujeme na rozpoznávání notových hlaviček, taktů a osnov, které jsou vyobrazeny na Obrázku 4.2.

Pro trénování modelu jsme zvolili dataset *OpenScore Lieder* [13, 14], obsahující tištěné noty. Obrázky s jejich veškerými anotacemi jsme získali díky softwaru

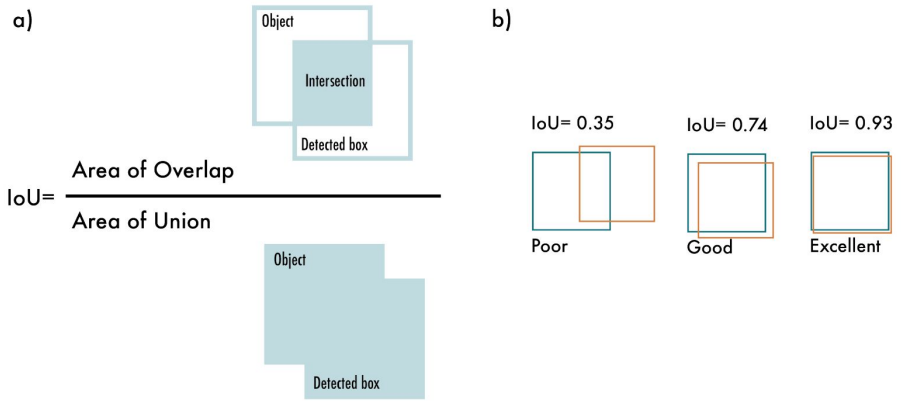


Obrázek 4.2 Ukázka vybraných objektů, jejichž detekci použijeme jako zástupnou úlohu pro vyhodnocení navržených postprocessingových metod. Notová hlavička zeleně, takt červeně a osnova modře.

MuseScore [15]. Následně jsme pomocí vlastních skriptů extrahovali a převedli do správných formátů potřebné informace pro náš model, čímž jsme získali trénovací data ve formě obrázků a anotací taktů a osnov.

Model YOLO (You Only Look Once) [12], zvolený pro tento experiment, je trénován na získaných připravených datech. YOLO byl vybrán pro úlohy detekce objektů na základě jeho rychlosti a efektivity. Natrénovaný model se vyhodnocuje na připraveném ručně anotovaném evaluačním datasetu. Ten je složen jak z not tištěných, tak z not ručně psaných, pro změření schopnosti modelu vyhodnocovat i typově neviděná data.

Výsledky modelu vyhodnocujeme pomocí metriky mAP50 (mean Average Precision at a 50% Intersection Over Union (IoU) threshold), která je standardní metrikou pro vyhodnocování úspěšnosti úlohy detekce objektů. Ukázka IoU viz Obrázek 4.3. Nezbytnou součástí trénovacího procesu je také validační dataset, který umožňuje zastavit trénování modelu ještě před tím, než dojde k jeho přetrénování (overfitting). Podobně jako dataset evaluační byl i tento dataset validační připraven a anotován ručně.



Obrázek 4.3 Intersection over Union (IoU). a) IoU se spočítá jako podíl velikosti průniku (intersection) bounding boxů - predikovaného a skutečného - a velikosti jejich sjednocení (union); b) příklady tří různých hodnost IoU pro tři různá umístění bounding boxů. [16]

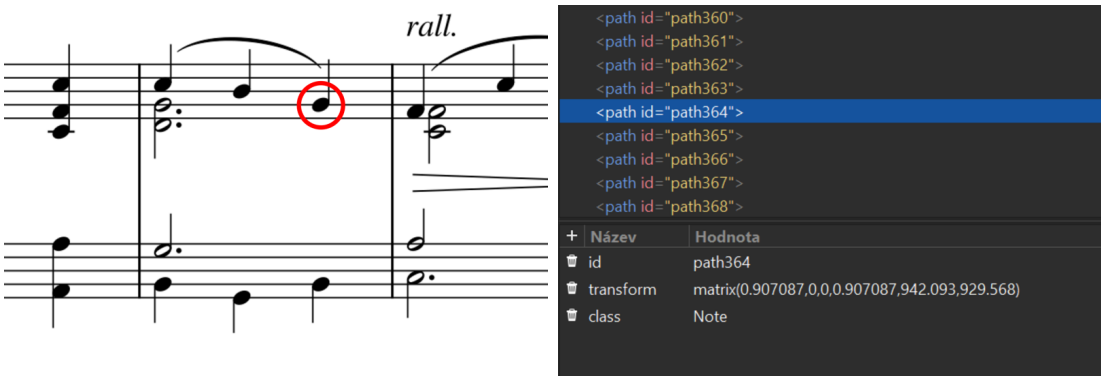
4.2 Datasety

4.2.1 Trénovací dataset

Jako trénovací dataset je použita kolekce digitálních notopisů OpenScore Lieder [13, 14]. Ta obsahuje piano noty ve formátu MuseScore (formát MSCX). Tento dataset se skládá z více než 5000 notopisů, tedy jednotlivých anotovaných obrázků a sestává zejména z kompozicí pro zpěv doprovázený klavírem. Na jednom obrázku se průměrně nachází kolem 200 notových hlaviček, asi 50 taktů a něco málo přes 10 osnov. Pro účel přípravy datasetu jsme využili funkcí programu MuseScore [15], které umožňují export z původního MSCX formátu do formátů PNG a SVG. Obrázek ve formátu PNG je rastrový obrázek složený z pixelů, stejně jako vstupní obrázky s notopisy určené k rozpoznávání. Soubory exportované do formátu PNG tak slouží jako trénovací obrázky. Obrázek ve formátu SVG je tzv. vektorový obrázek, tedy obrázek složený z různých geometrických útvarů, které se jednoduše analyzují nebo filtrují. Vzniklé SVG soubory jsou tak dále zpracovávány pro extrakci bounding boxů jednotlivých tříd a následného vzniku souborů anotací.

Anotace

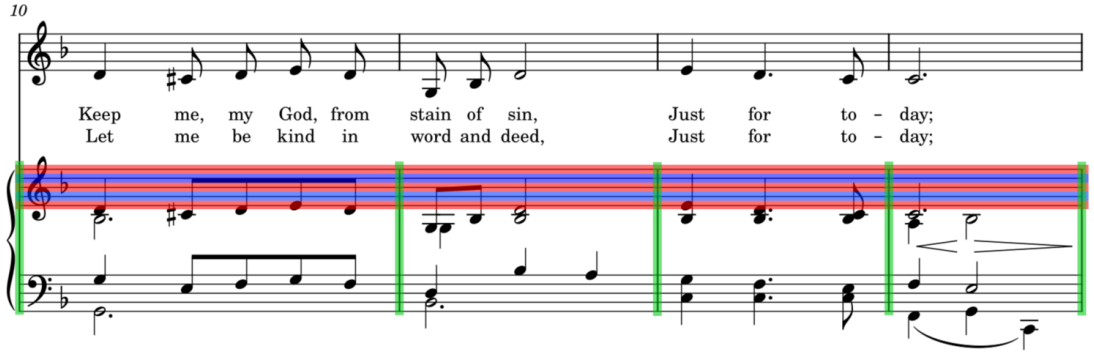
Aby byl model schopen naučit se rozpoznávat objekty na obrázcích, k těm trénovacím potřebuje mít navíc další soubor, kde jsou zaznačeny výskyty jednotlivých hledaných objektů v rámci obrázku. Těmito soubory jsou již zmíněné tzv. anotace. Každý rozpoznávaný objekt je ohraničen pomyslným obdélníkem a do anotačního souboru jsou zaznamenány rozměry a poloha na obrázku tohoto obdélníku. Pro vytvoření anotačního souboru ke každému jednomu obrázku v PNG formátu musíme extrahovat informace z vygenerovaného SVG souboru. Do anotací potřebujeme zahrnout informace o výskytu notových hlaviček, dále taktů a také osnov v rámci obrázku.



Obrázek 4.4 Objekt notové hlavičky s příkladem svého SVG výstupu z MuseScore [15]. Notovou hlavičku nalezneme pomocí anotace ”class: Note”, bounding box vypočítá knihovna svgelements [17] z transformační matice notové hlavičky.

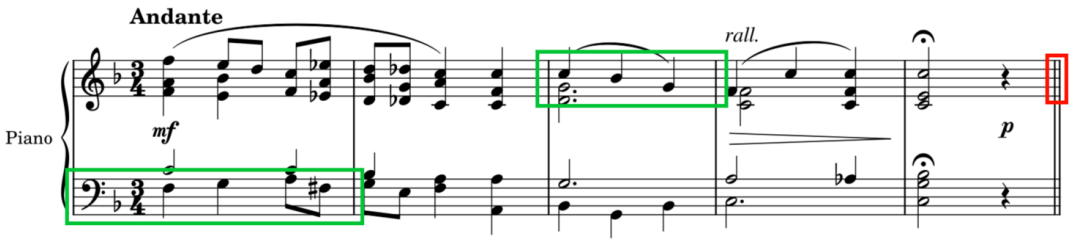
Námi připravený Python skript umí pomocí knihovny svgelements [17] jednoduše získat anotace notových hlaviček. Ty jsou totiž v SVG souboru reprezentovány pomocí jejich bounding boxu. Bounding box notové hlavičky sestává ze čtyř číselných informací, jedná se o záznam x-ové a y-ové souřadnice středu notové hlavičky v rámci obrázku a poté o šířku a výšku daného bounding boxu. Bounding box se tak získá z transformační matice a zároveň 2D cesty definující okraje objektu, kterou můžete vidět na Obrázku 4.4. Svgelements pak má k dispozici tuto křivku definující tvar notové hlavičky a z ní spočítá informace jejího ohraničujícího obdélníku. Při zisku anotací pro notové hlavičky tak knihovna svgelements [17] zkrátka vrátí patřičné bounding boxy.

Získávání anotací pro další dva typy námi rozpoznávaných objektů je pak o něco složitější. Objekty taktů a osnov neodpovídají jedna k jedné svému elementu v SVG reprezentaci. Oba typy objektů sestávají pouze z oddělených čar, jako je vidět na Obrázku 4.5. Tento problém řeší náš další skript v jazyce Python. V případě vytváření anotace osnov skript seřadí vzestupně podle jejich y-ové souřadnice veškeré linky odpovídající součástem osnovy. Následně je seskupí do pětic s kontrolou, jestli se nejedná o chybnou neúplnou osnovu. Poté vytvoří podobný bounding box, jako dostáváme u notových hlaviček, a to pomocí informací o první a poslední lince notové osnovy.



Obrázek 4.5 Vodorovné čáry, ze kterých sestávají osnovy a takty (červeně a modře) a svislé taktové čáry vytvářející jednotlivé takty (zeleně). Jednotlivé linky osnovy jsou označeny dvěmi barvami pouze z toho důvodu, aby na obrázku nesplývaly v jednolitou plochu.

Takty jsou horizontální úseky osnovy označující určitý časový úsek v hudbě. Jeden takt je vymezen dvěmi svislými čarami v rámci osnovy, jako jde vidět na Obrázku 4.6. K vytvoření anotací taktů náš Python skript využívá již zpracovaných osnov. Provedeme extrakci svislých taktových čar z SVG souboru a rozdělíme jimi připravené osnovy. Přičemž opět probíhá kontrola, jestli je daná taktová čára validním rozdělením osnovy a skutečně k dané osnově patří. Můžou totiž nastávat případy, kdy se jedna svislá taktová čára táhne přes několik osnov. Tak je tomu velmi často u skladeb s početnějším nástrojovým obsazením nebo klavírním partem obsahujícím 2 notové záznamy pro každou ruku zvlášť, jako je zvýrazněno zelenou barvou na Obrázku 4.5. Ona prodloužená taktová čára pak pomyslně spojuje nástroje a připomíná, že hrají současně. Výsledné informace pro vytvoření bounding boxů pro takty tak získáme z informací o připravené osnově, na které se takt nachází, a z ohraničujících čar taktu na osnově.

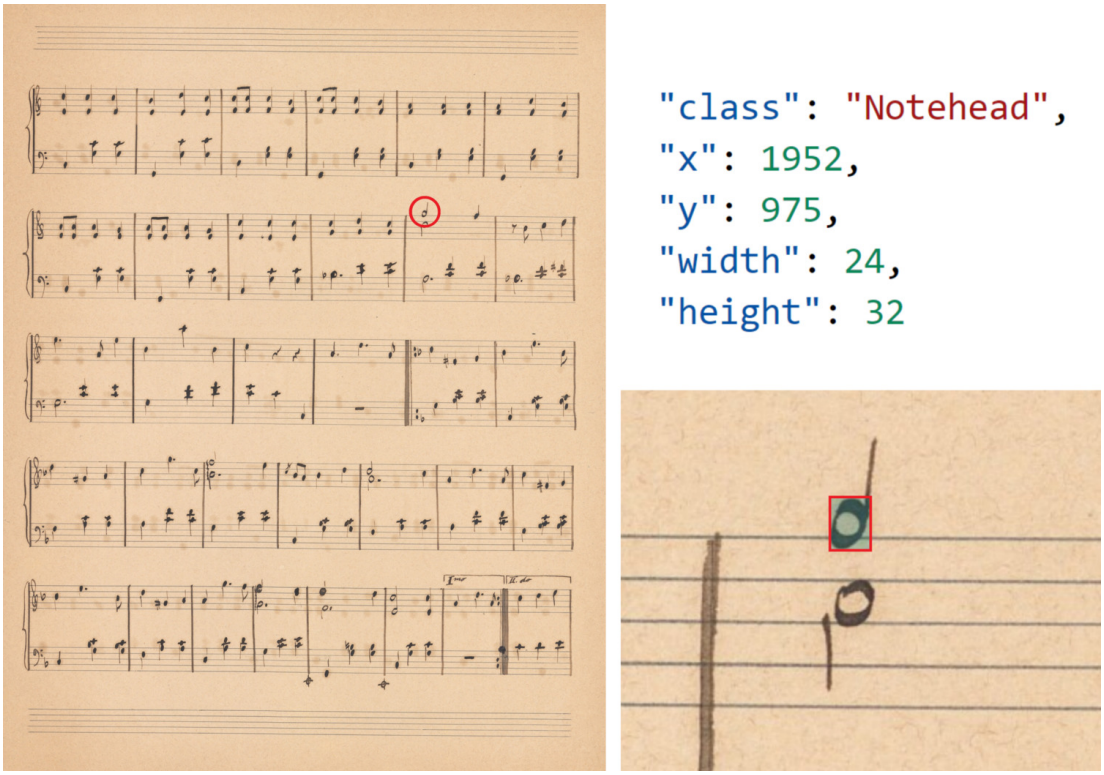


Obrázek 4.6 Takt je část osnovy ohraničená dvěma svislými taktovými čarami, ve které je hudební obsah (zeleně). Část osnovy ohraničená 2 svislými čarami bez dalšího hudebních obsahu zpravidla taktem není (červeně) - zde se jedná o tzv. "dvojčáru"označující konec skladby či její části.

4.2.2 Validační a evaluační dataset

Obrázky do validačního a evaluačního datasetu byly vybírány a anotovány ručně. Tato diverzita v trénovacím a validačním datasetu je klíčová pro ověření schopnosti modelu generalizovat a správně fungovat v různých kontextech. Z toho důvodu jak validační, tak evaluační dataset sestává napůl z not tištěných a napůl z not ručně psaných. Ručně psané noty pochází z volně dostupné části digitálního archivu Moravské zemské knihovny [8]. Noty tištěné jsou pak z archivu "International Music Score Library Project"známého pod zkratkou IMSLP [18].

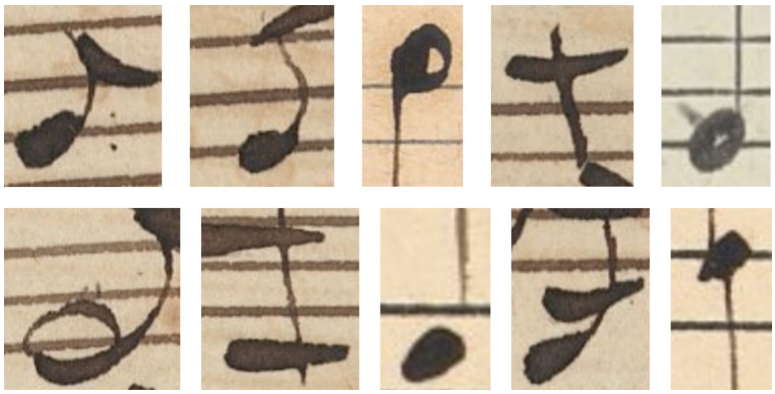
Anotace obrázků notopisů spočívají ve vyznačení bounding boxů jednotlivých symbolů stejně, jako je tomu u trénovacího datasetu. Problémem je, že ručně zjišťovat a zapisovat x-ovou a y-ovou souřadnici navíc s výškou a šířkou každého symbolu je prakticky nemožné. Jedná se o časově velmi náročnou činnost. Zjednodušení v tomto ohledu přinesl program Inkscape [19]. Jedná se o aplikaci na tvorbu a úpravu vektorových obrázků. Má řadu funkcí, pro nás však dvě nejdůležitější jsou přidávání vrstev k výchozímu plátnu a funkce nakreslení obdélníku. Můžeme tak vložit obrázek do aplikace, kolem každého jednoho symbolu nakreslit obdélník a vytvořit tím zmiňovaný bounding box. Pořád jde o časově náročnou práci, ovšem čas strávený anotováním je výrazně kratší, než by tomu bylo u jiného způsobu ruční anotace. Program Inkscape umí soubor převést do SVG formátu. Tuto funci využíváme a ze získaného SVG, podobně jako u trénovacího datasetu, extrahujeme připravené bounding boxy, viz Obrázek 4.7.



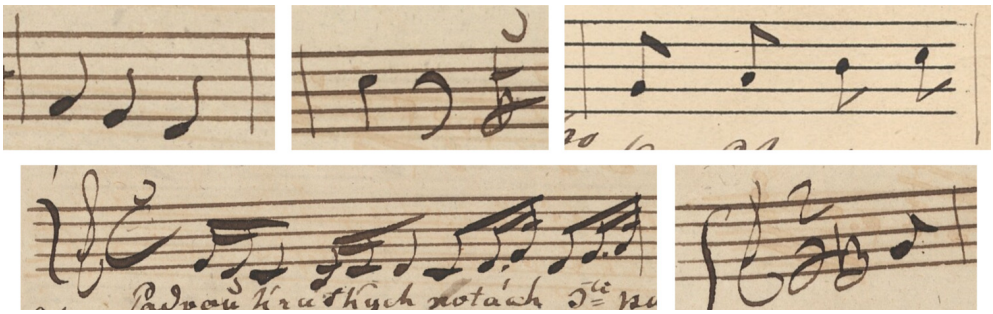
Obrázek 4.7 Ručně vyznačený bounding box notové hlavičky s později extrahovanými přesnými hodnotami šířky, výšky a souřadnic jejího levého horního rohu v pixelech (vpravo) se svým výskytem v rámci celého obrázku notopisu (vlevo).

4.3 Specifika ručně psaných notopisů

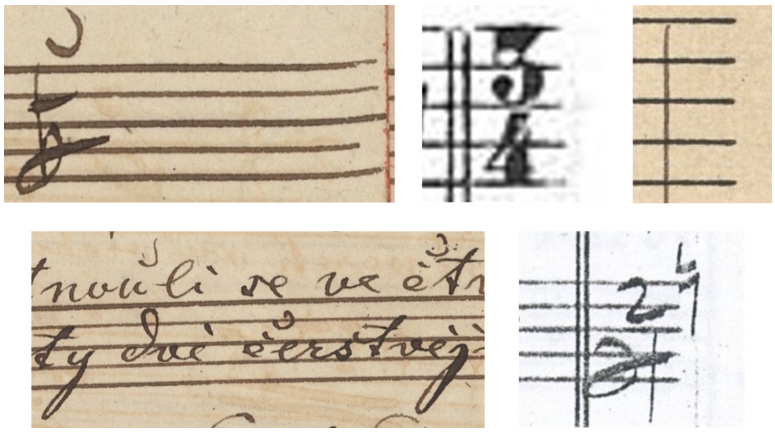
U ručně anotovaných not jsme objevili spoustu specifických vlastností tohoto typu notopisů. To potvrdilo nesmírné rozdíly s notopisy tištěnými a nutnost přistupovat k úloze rozpoznávání ručně psaných not s ohledem na jejich individuální charakteristiky a variabilitu. Na Obrázku 4.8 jsou vyobrazeny různé symboly, které však nesou ten jeden samý význam notové hlavičky. Podobně je tak tomu u symbolu taktu. Ten může vypadat různě, jak je uvedeno na Obrázku 4.9 a naopak toto, jako je na Obrázku 4.10 takty nejsou.



Obrázek 4.8 Deset různých symbolů nesoucích význam notové hlavičky.



Obrázek 4.9 Pět různých taktů.



Obrázek 4.10 Pět různých obrázků, které jsou velmi podobné taktům, a můžou tak být chybně identifikovány a označeny za takty.

4.4 YOLOv8

Ultralytics YOLO je série modelů, určených k řešení úlohy *object detection* neboli rozpoznávání objektů [12]. Verze 8, tedy model pod názvem YOLOv8, je pokročilý model k detekci objektů navržený pro úlohy vyžadující vysokou rychlost a přesnost. Je schopen zvládnout různé úlohy počítačového vidění, jak již zmíněnou detekci, tak i segmentaci, klasifikaci a další. Tato flexibilita ho činí vhodným pro rozmanité aplikace od jednoduchého sledování objektů po složité porozumění scéně [16]. Proto byl vybrán do této práce jako model pro provádění experimentů (kapitola 5) a následnou evaluaci funkčnosti a efektivity postprocessingu (kapitola 3). Využívaná schopnost modelu detekce objektů je přiblížena na Obrázku 4.1.

Knihovna ultralytics poskytující model YOLOv8 [12] umožňuje rozsáhlou přizpůsobivost modelu během tréninku a podporuje různé hyperparametry nebo režimy tréninku. Snadno lze specifikovat cesty k datovým sadám, délku tréninku a téměř jakékoliv jiné parametry, které umožňují pohodlnou práci. Další výhodou použití YOLOv8 modelu je fakt, že trénink modelu končí ve chvíli, kdy validační chyba dosáhne svého minima. A to bez ohledu na celkový plánovaný počet epoch - knihovna ultralytics [12] tak poskytuje tzv. *early stopping funkci*, která zabraňuje přetrénování. Po natrénování umí model YOLOv8 [12] zpracovat a vyhodnocovat vstupy obrázků i videí. V naší práci nás zajímají zejména výsledky trénovaní YOLOv8, které spočívají ve schopnosti identifikace objektů a přiřazení jejich třídy. Výsledné detekované objekty jsou pak ohraničeny rámečky s příslušným kódem třídy.

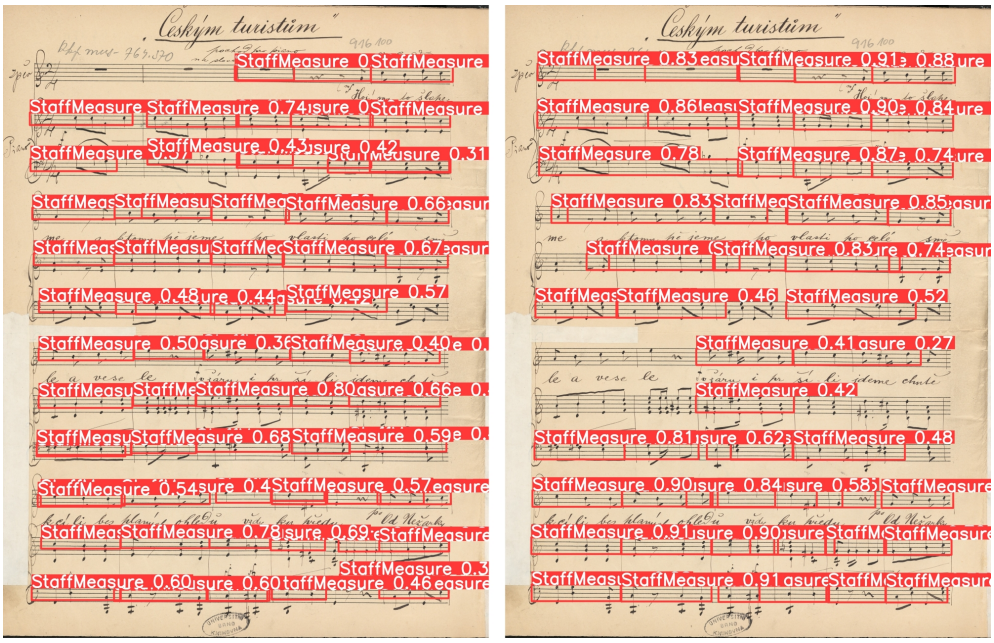
Model YOLOv8 [12] je schopen zpracovat několik úloh současně, konkrétně při jednom průchodu sítí detekovat a klasifikovat všechny bounding boxy najednou a ne každý zvlášť. Odtud nese i svůj název *You Only Look Once*. Model je navržen tak, aby identifikoval a klasifikoval různé objekty v obraze s vysokou přesností a rychlostí. Primární výhodou modelu je zmíněná rychlost [20]. Rovněž je známý svou schopností provádět detekce téměř v reálném čase s vysokou úrovní přesnosti.

Pro trénování modelu YOLOv8 [12] implementuje knihovna ultralytics i standardní techniky data augmentation. Tyto techniky zahrnují různé transformace dat, jako jsou změny měřítka, oříznutí nebo rotace obrázků, což pomáhá modelu lépe generalizovat u neviděných dat a rovněž zabraňuje jeho přetrénování. Díky tomu není nutné tyto procesy implementovat ručně, což šetří čas a zjednodušuje proces tréninku. Z hlediska implementace se tak můžeme soustředit na námi vybrané augmentace dat blíže popsané v kapitole 3. Při tréninku modelu tak dochází ke spojení námi implementovaných postprocessingových metod degradací s nativními augmentacemi knihovny ultralytics [12].

5 Experimenty

5.1 Popis experimentů

V rámci experimentů byl YOLO model trénován výhradně na datasetu složeného z not tištěných. Na tento dataset byly postupně aplikovány implementované degradace blíže popsané v kapitole 3. Navzdory absenci ručně psaných not v trénovacím datasetu model dosahuje dobrých výsledků i na těchto notopisech, což naznačuje určitou míru generalizace. Na Obrázku 5.1 se nachází srovnání predikcí modelů, kde první model (jeho predikce vlevo) byl natrénován na základním trénovacím datasetu bez žádné degradace. Oproti tomu druhý obrázek (vpravo) je predikce modelu, jehož trénovací data prošla všemi metodami postprocessingu, tedy data byla degradována pomocí všech operací zmíněných v kapitole 3.

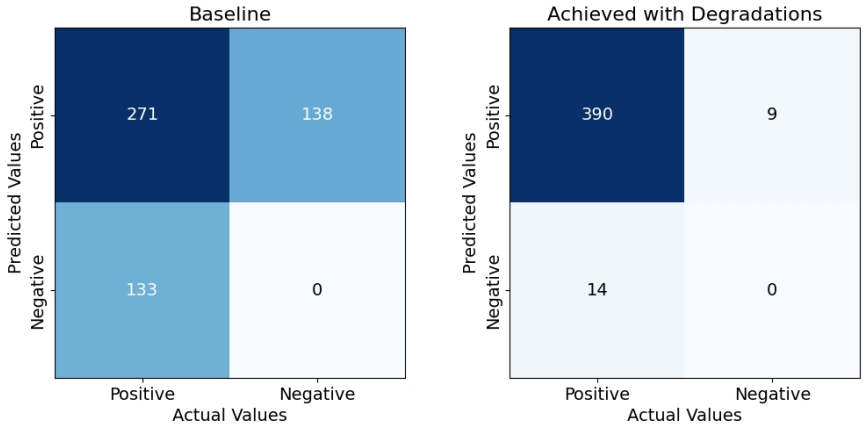


Obrázek 5.1 Výrazně méně úspěšná predikce modelu na ručně psaných notách. Model byl však trénován na notopisech tištěných, ovšem bez jakékoliv postprocessingové degradace (vlevo). Úspěšná predikce modelu na ručně psaných notách. Model byl však natrénován na notopisech tištěných, ovšem se všemi postprocessingovými degradacemi (vpravo).

Uvedené predikce na Obrázku 5.1 obsahují ručně psaný notopis z evaluačního datasetu, na kterém je červenými obdélníky vyznačena predikce modelu v podobě určení místa výskytu jednotlivých taktů. U každé predikce je rovněž uvedeno desetinné číslo v rozmezí 0 až 1 popisující, s jakou pravděpodobností se jedná o daný objekt. Čím více se blížíme hodnotě 1, tím si je model jistější svou predikcí. Při porovnání obrázků vidíme, že model natrénovaný na datech s postprocessingem (vpravo) sice detekuje méně taktů, ale jeho predikce jsou mnohonásobně přesnější. Níže, v sekci 5.1.2, po vysvětlení tohoto jevu pomocí metrik precision a recall seznáme, že i přes na první pohled nepěknou vlastnost chybějících predikcí můžeme zcela jasně potvrdit, že námi implementované degradace jednoznačně zlepšily přesnost modelu v úloze rozpoznávání objektů.

5.1.1 Matice záměn

Na obrázku 5.2 vidíme *matice záměn* (z ang. confusion matrix, jinak také matice zmatení nebo konfuzní matice) modelů pro úlohu rozpoznávání objektů taktů. Tyto matice záměn poskytují detailní pohled na výkonnost modelu při klasifikaci. V levé horní části se nachází položka *true positive*, tedy reálné objekty taktů, které byly skutečně správně klasifikovány jako takty. V pravé horní části se nacházejí objekty *false positive*, které sice byly klasifikovány jako takty, ale o tento objekt se nejedná. V levé dolní části jsou *false negative* objekty, tedy skutečné takty, které ovšem nebyly úspěšně detekovány, a nakonec v pravém dolním rohu *true negative* jsou položky správně klasifikované tak, že se nejedná o námi hledaný objekt.



Obrázek 5.2 Vlevo je matice záměn (z ang. confusion matrix) modelu natrénovaného na základním trénovacím datasetu bez degradací. Model správně klasifikoval 271 objektů. Dalších 138 objektů bylo nesprávně detekováno a označeno za hledaný objekt. Navíc 133 objektů, které měly být klasifikovány jako hledaný objekt, nebylo detekováno vůbec. Tento výsledek naznačuje nižší hodnotu presicion (0,66) i recall (0,67) modelu bez aplikace postprocessingových degradací. Vpravo je matice záměn modelu, jehož trénovací data prošla všemi metodami postprocessingu uvedenými v kapitole 3. Model správně klasifikoval 390 objektů a nesprávně klasifikoval pouze 9 objektů. Naopak, pouze 14 objektů se nepodařilo detekovat. Tento výsledek demonstruje výrazné zlepšení hodnot precision (0,98) i recall (0,97) modelu díky použití postprocessingu a augmentací dat. Všechny hodnoty precision a recall jsou spočítané na evaluačním datasetu.

Obrázek 5.2 znázorňuje dvě matice záměn. Vlevo je vyobrazena matice záměn modelu, který byl trénován na základním trénovacím datasetu bez jakékoli degradace. Z této matice je patrné, že model měl potíže s přesnou klasifikací, což se projevuje vysokým počtem nesprávných klasifikací. Konkrétně, model správně klasifikoval 271 objektů jako takt, ale 138 objektů bylo klasifikováno jako takt nesprávně, jedná se o jiné objekty. Na druhou stranu, 133 objektů taktů nebylo vůbec detekováno. Hodnota precision tohoto modelu je 0,66. Recall u tohoto modelu nabývá hodnoty 0,67.

Na Obrázku 5.2 se vpravo nachází matice záměn modelu, jehož trénovací data prošla všemi metodami postprocessingu, data tedy byla degradována pomocí všech operací zmíněných v kapitole 3. Výsledky tohoto modelu jsou výrazně lepší, což je patrné z nízkého počtu nesprávných klasifikací. Model správně klasifikoval 390 objektů a v množině *false positive* a *false negative* skončilo dohromady pouze 23 objektů. Hodnota precision zde dosáhla 0,98 a hodnota recall 0,97. Z porovnání

všech hodnot je tedy zřejmé, že implementované degradace výrazně zlepšily jak precision, tak recall modelu v úloze rozpoznávání objektů. Tato skutečnost potvrzuje, že použití postprocessingu a augmentací dat má pozitivní vliv na schopnost modelu generalizovat a správně klasifikovat i objekty, které nebyly přímo zahrnuty v trénovacím datasetu.

5.1.2 Precision a Recall

Pro kontrolu kvality a přesnosti modelů se nejčastěji používají dvě evaluační metriky. Jedná se o *precision* a *recall*. Hodnoty těchto metrik můžeme spočítat právě díky maticím záměn. Hodnota precision je definovaná jako podíl počtu prvků true positive a součtu true positive s false positive. V kontextu rozpoznávání objektů taktů nám precision říká, jaká část objektů, které model identifikoval jako takty, jsou skutečně takty. Vysoká hodnota precision tedy znamená, že model má málo falešně pozitivních předpovědí, což je nezbytné pro aplikace, kde je důležitá vysoká přesnost identifikace hledaných objektů.

Hodnota metriky recall je definovaná jako podíl počtu prvků true positive a celkového počtu skutečných pozitivních prvků (true positive + false negative). V kontextu rozpoznávání objektů taktů nám tak recall říká, jaká část skutečných objektů taktů byla modelem správně identifikována. Vysoká hodnota recall tedy znamená, že model dokáže odhalit většinu (nebo všechny) skutečných objektů taktů, což je důležité pro aplikace, kde je klíčové minimalizovat počet chybějících (false negative) detekcí. Zatímco precision se zaměřuje na to, jak přesné jsou pozitivní předpovědi modelu, recall se zaměřuje na to, jak úplné jsou pozitivní předpovědi.

5.1.3 Ablační analýza

Pro podrobnější analýzu vlivu jednotlivých degradací jsme použili tzv. *ablační analýzu* [21]. Ablační analýza je metodologický přístup používaný k hodnocení významu jednotlivých složek systému, v našem případě jednotlivých degradací aplikovaných na dataset, na celkovou výkonnost modelu. Cílem je identifikovat, jakou mají různé komponenty přínosnost pro dosažení optimálních výsledků.

V rámci našich experimentů jsme použili ablační analýzu následujícím způsobem:

- Základní experiment: Nejprve jsme natrénovali model na datasetu bez jakýchkoli degradací, což nám poskytlo základní výchozí bod pro porovnání.
- Aplikace všech degradací: Poté jsme natrénovali model na datasetu, na který byly aplikovány všechny implementované degradace. Tento krok nám umožnil zjistit, jaké maximální zlepšení výkonnosti můžeme dosáhnout použitím všech dostupných degradací.
- Jednotlivé ablace: V následných experimentech jsme postupně jednotlivé degradace vynechávali, zatímco ostatní degradace byly stále aplikovány. Pro každý takový experiment jsme změřili mAP, abychom zjistili, jak se odstranění konkrétní degradace projeví na výkonnosti modelu.

Tímto způsobem jsme mohli kvantifikovat příspěvek každé jednotlivé degradace a zjistit, která z degradací má největší dopad na výkonnost modelu.

5.2 Problémy s konvergencí

5.2.1 Problém objektu notové hlavičky

Přestože notové hlavičky představují zásadní část notového záznamu, jejich příliš malá velikost a poměrně nízké rozlišení způsobilo problémy v detekci. Samotné rozpoznávání objektů však není pro naši práci podstatné, a proto byly notové hlavičky z množiny rozpoznávaných objektů vyřazeny. Taktové čáry a osnovy jsou naopak rozpoznávány s výrazně větší přesností, a tak mohou sloužit k měření úspěšnosti metod postprocessingu.

Jak již bylo zmíněno, původně byly vybrány tři typy objektů pro testování metod postprocessingu na úloze rozpoznávání objektů. Jedná se o notové hlavičky, takty a osnovy vyobrazeny na obrázcích 1.2, 1.3, 1.4 a 1.5. Problém nastal u detekce notových hlaviček. Jedna notová hlavička zabírá jen přibližně 0,005 % plochy jednoho obrázku. Doporučené vstupní rozlišení obrázků pro trénink modelu YOLOv8 je 640 pixelů v jednom z rozměrů. To znamená, že jedna notová hlavička má šířku 5-6 pixelů, výšku 3-4 pixely a celá zabírá na obrázku nejvýše kolem 20 pixelů z celkových nejméně 400 000 pixelů obrázku. V této situaci je pro model skoro až nemožné určit výskyt takto drobného objektu.

Za účelem zlepšení šance modelu rozpoznat malé objekty notových hlaviček jsme vyzkoušeli dva různé přístupy modifikace trénovacího datasetu. Prvním z nich byla práce s bounding boxem neboli ohraničením výskytu notové hlavičky. Zmenšení bounding boxu pro naši situaci nemá smysl, jelikož problém je s příliš malými rozpoznávanými objekty. Bounding box musíme tedy jedině zvětšit. Po této operaci však může nastat jev, kdy se v bounding boxu notové hlavičky nacházejí i jiné objekty nebo jejich části. Vyzkoušeli jsme různé míry zvětšení bounding boxu. Nejlepších výsledků dosahoval model při dvojnásobném zvetšeni bounding boxu, ovšem stále se jednalo o obecně velmi špatný výsledek rozpoznání objektů. Je to způsobeno tím, že i přes zvětšení bounding boxu, díky kterému notová hlavička zabírá až 0,01 % obrázku, je rozpoznávaný objekt vůči celému obrázku stále velmi malý. Modely s většími bounding boxy podávaly ještě horší výsledky z důvodu ztráty důležitosti daného hledaného objektu v rámci příliš velkého vymezeného prostoru, kde se objekt nachází. Tato metoda tak bohužel nefungovala.

K podobnému závěru dospěje i druhá metoda řešení problému příliš malých objektů notových hlaviček. Touto metodou je zvětšení rozlišení trénovacích obrázků. Model by tak místo obrázků s výchozím rozlišením 640 pixelů mohl dostat obrázky například s 1280 pixely. Poměrově se však nic nezměnilo. Notová hlavička, nyní zabírající klidně už 100 pixelů, je vůči celému obrazu stále jen 0,01 % jeho obsahu.

Přirozeně navazujícím řešením tohoto problému je změnit poměr velikosti notové hlavičky vůči velikosti celého obrázku. Metoda funguje na základě rozdělení vstupního obrázku na menší části se zachováním informací na jednotlivých částech obrázku. Pokud například obrázek rozpůlíme, notové hlavičky budou rázem zabírat ne 0,01 % celkové plochy, ale už 0,02 %. Při dalším půlení zabírají objetky i 0,04 % plochy. Takovýmto dělením se můžeme dostat ke stavu, kdy notová hlavička zabírá nějakou minimální plochu dostatečně velkou na to, aby mohla být modelem rozpoznána. Problémem však je, že výsledný kus obrázku vypadá již úplně jinak než původní celý obrázek. Na začátku máme celý notopis, který obsahuje třeba 20 osnov, 100 taktů a 500 nerozpoznatelných notových hlaviček. Půlením můžeme

dosáhnout stavu obrázku s rozpoznatelnými notovými hlavičkami, ale už jen s půlkou osnovy a třeba dvěma takty. Tento stav již není notopisem, ale jen vyobrazením pár hudebních symbolů. Nemůžeme tak provádět experimenty této práce, tj. zkoumání efektu postprocessingu syntetických notopisů v kontextu jejich rozpoznávání, a musíme notovou hlavičku z rozpoznávaných objektů vyřadit.

Pokud bychom chtěli nalézt řešení pro problém notových hlaviček, využili bychom tohoto "stříhání"obrázku. Následně po detekci dostatečně velkých notových hlaviček bychom sešili nastříhaný obrázek zpět do jednoho celku. Tato metoda by však znamenala, že by notové hlavičky musely být detekovány odlišným systémem než ostatní objekty, čímž by se staly speciálními případy. Hlavním důvodem, proč jsme tuto metodu nezvolili, je náročnost na úpravu trénovacího kódu, pro kterou nebyl dostatek času. Pokud bychom však v budoucnu chtěli vytvořit specifický detektor hlaviček, tato cesta by byla jednou z možných řešení.

5.2.2 Problém objektu osnovy

Rozpoznávání notových osnov se ukázalo jako obtížnější úkol ve srovnání s rozpoznáváním taktů. Většina modelů byla natrénována úspěšně, plní svůj účel a umí rozpoznávat objekty osnov. Jiné modely však nezkonvergovaly. Důvodů nedosažení konvergence u modelů pro osnovy je několik. Jedním z klíčových faktorů je rozdílný počet instancí těchto dvou tříd na tréninkových datech. Na jednu osnovu připadá několik taktů, je jich tedy na jedné straně notopisu výrazně více než notových osnov. Model tak má k dispozici několikanásobek trénovacích příkladů pro takty oproti osnovám, a to může vést k lepší generalizaci modelu pro takty ve srovnání s notovými osnovami.

Dalším důležitým faktorem je vzhled notových osnov. Jejich šířka je někdy až desetkrát větší než jejich rozměr na výšku, což může rovněž způsobovat problémy při detekci. Na rozdíl od taktů, které nemají tak výrazný rozdíl mezi rozměry svých výšek a šířek, osnovy zabírají výrazně více místa na stránce, a můžeme je tak zařadit do skupiny velkých objektů. Vzhledem k tomu, že model YOLO je navržen pro detekci spíše menších a středně velkých objektů, může mít problémy s rozpoznáváním objektů s neobvyklými tvary nebo velikostmi, a právě to vede k horší přesnosti při detekci osnov.

Řešením problému nedostatečné generalizace modelu pro notové osnovy by bylo v první řadě navýšení počtu trénovacích dat. Dále bychom mohli zkusit úpravu modelu nebo použití alternativních technik specificky navržených pro rozpoznávání větších protáhlých objektů. Vzhledem k povaze naší práce a experimentů spočívajících ve zkoumání účinků postprocessingu však není nutné zástupnou úlohu ve formě rozpoznávání objektů dále rozvíjet. K vyhodnocení experimentů s notovými osnovami použijeme všechny modely takové, které zkonvergovaly.

5.3 Výsledky experimetů

Celkové výsledky experimentů jsou shrnuty v tabulkách 5.1 a 5.2. Z analýzy tabulek je patrné, že aplikace jednotlivých metod postprocessingu měla výrazný dopad na přesnost modelu v úloze rozpoznávání objektů. Největší zlepšení v přesnosti rozpoznávání taktů bylo dosaženo aplikací všech degradací. Zlepšení bylo pozorováno také u modelů, které prošly pouze jednotlivými degradacemi zvlášť. Došlo tak k potvrzení účinnosti našich postprocessingových technik.

Použité degradace	**mAP pro osnovy**	**mAP pro takty**
Žádné	0,916	0,661
Všechny	0,257	0,991
Všechny kromě textury	0,150	0,889
Všechny kromě prosaku	0,995	0,984
Všechny kromě rukopisu	0,995	0,989
Všechny kromě šumu	0,995	0,977

Tabulka 5.1 Výsledky experimentů - hodnota mAP50 pro validační dataset

Jak již bylo blíže popsáno v kapitole 4.4, při trénování dochází k jevu zvanému early stopping. V našem případě docházelo k úspěšnému natrénování jednotlivých modelů typicky kolem čtrnácté až sedmnácté epochy.

Použité degradace	**mAP pro osnovy**	**mAP pro takty**
Žádné	0,808	0,869
Všechny	0,400	0,921
Všechny kromě textury	0,423	0,906
Všechny kromě prosaku	0,929	0,954
Všechny kromě rukopisu	0,944	0,931
Všechny kromě šumu	0,287	0,954

Tabulka 5.2 Výsledky experimentů - hodnota mAP50 pro evaluační dataset

6 Závěr

Cílem práce bylo definovat, implementovat a změřit techniky syntézy trénovacích dat, které by pomohly v úloze Optical Music Recognition (OMR). Testování úspěšnosti navržených metod v kapitole 3 proběhlo na zástupné úloze rozpoznávání objektů (hudebních symbolů). V rámci experimentů, blíže popsaných v kapitole 5, byl model YOLO trénován na datasetu složeném výhradně z tištěných not. Na tento dataset byly postupně aplikovány implementované degradace. Navzdory absenci ručně psaných not v trénovacím datasetu model dosahuje dobrých výsledků i na těchto notopisech, což naznačuje úspěšnost generalizace.

Tato práce demonstrovala, že použití definovaných metod postprocessingu na trénovací data může výrazně zlepšit schopnost modelů generalizovat a přesně klasifikovat objekty na notopisech. Dá se tak očekávat, že výsledky práce zlepší výkonnost modelů i na úloze rozpoznávání hudebního obsahu notopisu. Naše poznatky mohou být užitečné i pro budoucí výzkum a vývoj algoritmů na rozpoznávání hudebních notací včetně dalších jiných úloh oblasti OMR. Konkrétní aplikací této práce bude její začlenění do vznikajícího projektu Mashcima 2, navazujícího na původní práci Mashcima [4].

Literatura

1. Calvo-Zaragoza, Jorge; Hajič jr., Jan; Pacha, Alexander. Understanding Optical Music Recognition. *ACM Computing Surveys*. 2020, roč. 53, č. 4, s. 77. Dostupné z doi: 10.1145/3397499.
2. Tuggener, Lukas; Satyawan, Yvan Putra; Pacha, Alexander; Schmidhuber, Jürgen; Stadelmann, Thilo. The DeepScoresV2 dataset and benchmark for music object detection. In: *25th International Conference on Pattern Recognition (ICPR)*. Online, 2020, s. 9188–9195. Dostupné z doi: 10.1109/ICPR48806.2021.9412290.
3. Calvo-Zaragoza, Jorge; Rizo, David. End-to-End Neural Optical Music Recognition of Monophonic Scores. *Applied Sciences*. 2018, roč. 8, č. 4. issn 2076-3417. Dostupné z doi: 10.3390/app8040606.
4. Mayer, Jiří; Pecina, Pavel. Synthesizing Training Data for Handwritten Music Recognition. In: *16th International Conference on Document Recognition and Analysis*. Lausanne, Switzerland, 2021, s. 626–641. Dostupné z doi: 10.1007/978-3-030-86334-0_41.
5. Hajič, jr., Jan; Pecina, Pavel. The MUSCIMA++ Dataset for Handwritten Optical Music Recognition. In: *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Kyoto, Japan, 2017, s. 39–46. Dostupné z doi: 10.1109/ICDAR.2017.16.
6. Efros, A. A.; Freeman, W. T. Image Quilting for Texture Synthesis and Transfer. *28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. 2001, s. 341–346.
7. Efros, Alexei A.; Leung, Thomas K. Texture Synthesis by Non-parametric Sampling. 1999, s. 1033–1038.
8. *Digitální archiv Moravské zemské knihovny* [online]. [cit. 2024-04-25]. Dostupné z: https://www.digitalniknihovna.cz/mzk.
9. Kanungo, T.; Haralick, R. M.; Baird, H. S.; Stuezle, W.; Madigan, D. A statistical, nonparametric methodology for document degradation model validation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2000, roč. 22, č. 11, s. 1209–1223. Dostupné z doi: 10.1109/34.888707.
10. Jr., Jan Hajič; Pecina, Pavel. *Detecting Noteheads in Handwritten Scores with ConvNets and Bounding Box Regression*. 2017. Dostupné z arXiv: 1708.01806 [cs.CV].
11. Pacha, Alexander. Incremental supervised staff detection. In: *Proceedings of the 2nd international workshop on reading music systems*. 2019, s. 16–20.
12. *Ultralytics YOLOv8 Docs* [online]. [cit. 2024-05-04]. Dostupné z: https://docs.ultralytics.com/.
13. Gotham, Mark; Jonas, Peter; Bower, Bruno; Bosworth, William; Rootham, Daniel; VanHandel, Leigh. Scores of scores: an openscore project to encode and share sheet music. In: *5th International Conference on Digital Libraries for Musicology (DLfM)*. Paris, France, 2018, s. 87–95. Dostupné z doi: 10.1145/3273024.3273026.

14. Gotham, Mark Robert Haigh; Jonas, Peter. The OpenScore Lieder Corpus. In: *Music Encoding Conference*. Alicante, Spain, 2022, s. 131–136. Dostupné z DOI: 10.17613/1my2-dm23.

15. *MuseScore* [online]. [cit. 2024-05-18]. Dostupné z: https://musescore.com/.

16. Terven, Juan; Córdova-Esparza, Diana-Margarita; Romero-González, Julio-Alejandro. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction*. 2023, roč. 5, č. 4, s. 1680–1716. Dostupné z DOI: 10.3390/make5040083.

17. *Project description svgelements* [online]. [cit. 2024-05-18]. Dostupné z: https://pypi.org/project/svgelements/.

18. *IMSLP* [online]. [cit. 2024-05-18]. Dostupné z: https://imslp.org/.

19. *Inkscape* [online]. [cit. 2024-05-18]. Dostupné z: https://inkscape.org/.

20. Huang, Jonathan; Rathod, Vivek; Sun, Chen; Zhu, Menglong; Korattikara, Anoop; Fathi, Alireza; Fischer, Ian; Wojna, Zbigniew; Song, Yang; Guadarrama, Sergio; Murphy, Kevin. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In: *30th Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, Hawai, USA, 2017, s. 3296–3297. Dostupné z DOI: 10.1109/CVPR.2017.351.

21. Meyes, Richard; Lu, Melanie; Puiseau, Constantin Waubert de; Meisen, Tobias. *Ablation Studies in Artificial Neural Networks*. 2019. Dostupné z arXiv: 1901.08644 [cs.NE].