OCR pro notové zápisy

Marek Salát [xsalat00@stud.fit.vutbr.cz](mailto:xsalat00@stud.fit.vutbr.cz)

Matěj Dohnal [xdohnal28@stud.fit.vutbr.cz](mailto:xdohnal28@stud.fit.vutbr.cz)

Richard Pánek [xpanek06@stud.fit.vutbr.cz](mailto:xpanek06@stud.fit.vutbr.cz)

# Zadání

Implementujte OCR metodu pro přepis tištěných not. Můžete předpokládat, že jsou obrazy stránek velmi dobře pořízené a že nejsou stránky geometricky deformované. Přesnost rozpoznání vyhodnoťte na vhodné sadě obrázků. Zaměřte se jen na vhodnou podmnožinu notace.

Možný postup:

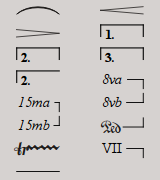
* detekce notové osnovy - Například pomocí Hough transform nebo sumace pixelů v horizontálním směru a hledání minim intenzit.
* odstranění linek notové osnovy při zachování celistvosti ostatních objektů
* segmentace objektů - adaptivní prahování a nalezení spojitých komponent
* extrakce příznaků ze segmentovaných útvarů
* klasifikace segmentů na základě extrahovaných příznaků a určení tónů podle pozice

# Omezení

Návrh projektu vychází z předpokladu, že obrázek je dokonale rovný (pořízen například z převodu PDF dokumentů). Dále pro zjednodušení neuvažujeme, některá značení a to:

* Akordy



* Rytmické notové skupiny, tedy noty spojené v taktu, tak i noty spojené přes více řádků (mezi klíči). Obecně i propojení více řádků, například závorkami a čárami  
  
* Notové zápisy bubnů a jiných specifických nástrojů
* Linky  
  
* Svorky a obecně doplňující text (slova písně, tempo atd.) nad a pod osnovou
* Legata  
  
* Ligatury  
  
* Hlasy   
  Hlasy 1 a 2
* Volta  
  

# Návrh a implementace

Obraz je nejdříve převeden na černobílý metodou OTSU. Poté jsou odstraněny linky notové osnovy. Osnova slouží člověku k odhadnutí výšky tónu, ale ve zpracování obrazu přináší problémy v segmentaci symbolů. S osnovou nemůžeme jednoduše oddělit jednotlivé symboly, protože jimi prochází linky. Pokud je osnova odstraněna, můžeme jednoduše segmentovat jednotlivé symboly pro další zpracování. Výška tónu může být odhadnuta podle hlavičky noty, pozice odstraněných linek a výšky řádku. Jednotlivé symboly klasifikujeme postupem popsaným níže v textu. U symbolů které obsahují notu, je výška tónu odhadnuta na základě pozice hlavičky. Celá nota rozpoznána detektorem hlaviček, stejně tak nota půlová, která navíc obsahuje stonek (je výrazně vyšší než širší). Další typy not, respektive noty s černou hlavičkou, které se liší pouze počtem praporů, jsou rozpoznány detektorem symbolů.

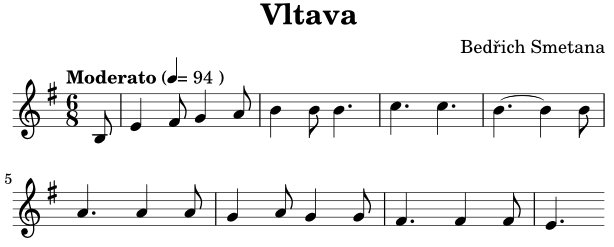
## Odstranění notové osnovy

Před odstraněním not, je nutné nejdříve detekovat linky osnovy. Jelikož jsme vycházeli z předpokladů, že obraz je rovný, bylo možné použít následující postup. Nejdříve jsou vytvořeny průměty osy y (počet černých pixelů pro jednotlivé řádky) viz. následující obrázek.

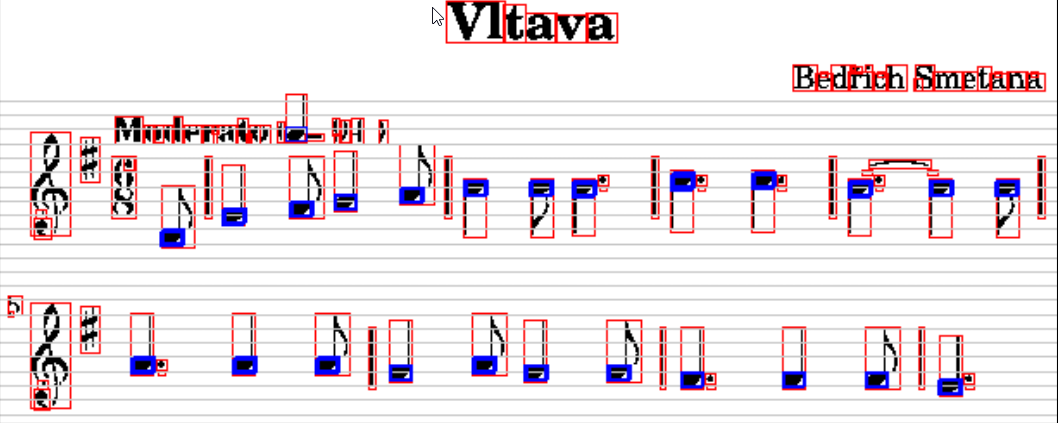


Liky vytvoří špičaté vrcholy. Za linku se považuje ten vrchol, který je větší než 80% šířky. Šírka linky je vypočtena jako průměrná šířka vrcholů v celém obrázku. Výška linky je vypočten jako průměrná výška řádku mezi pěti linkami (neuvažuje se mezera mezi osnovami). Hledání osnovy, šířky linek a výšky řádku je implementováno v staff/staff\_finder.py:StaffFinder

Následuje odstraňování linek. Pro každý vrchol (linku) a každou pozici ve vertikálním směru se hledá nepřerušený sled černých pixelů v horizontálním směru. Pokud je tento sled menší než 110% šířky linky, je nahrazen bílými pixely.

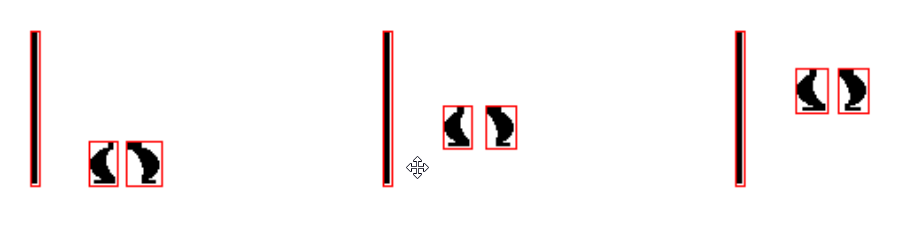


Obrázek 1. Před zpracováním

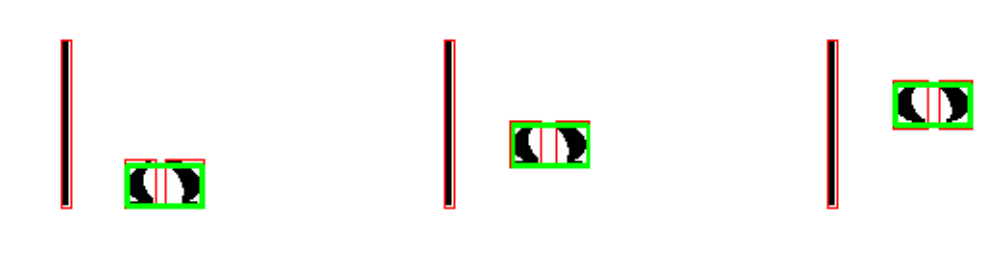


Obrázek 2. Po odstranění linek (linky jsou naznačeny šedě). Červené čtverce uzavírají oblast deekovaných symbolů. Modré čtverce zvýraňují detekovaneé hlavíčky

Linky jsou odstraňovány i pro pomocné linky (tzv. žebříky) nad a pod osnovou, maximálně však 4 nad a 4 pod. U tohoto typu mazání linek je nevýhoda, že výsledný obraz může být mírně deformován. Také dochází k půlení celých a k částečnému půlení půlových not, které leží mezi linkami viz. následující obrázek.



Jelikož jsou hlavičky hledání v originálním obrázku, tak nám toto chování nevadilo, protože pokud je symbol uvnitř hlavičky, tak může být zahozen. Následuje obrázek se zvýrazněnou hlavičkou po detekci hlaviček.



Odstraňování linek je implementováno v staff/staff\_remover.py:StaffRemover

## Segmentace symbolů

Segmentace symbolů je naimplementována v staff/symbol\_extractor.py: SymbolExtractor za pomocí OpenCV funkce cv2.findContours. Následuje pouze transformace hierarchie kontur a děr, tak že kontury nebo díry, které vzájemně koliduji, jsou spojeny a je vytvořena nová obalující kontura.

## Hledání hlaviček not

Detekce hlaviček je prováděna na originálním obrázku, aby bylo možné detekovat i hlavičky celých a půlových not (odstranění linek je může rozpůlit). Je implantována metodou *template matching* s prahem 0.65 za pomocí OpenCV (cv2.matchTemplate(segment, self.template, cv2.TM\_CCOEFF\_NORMED)). Šablony jsou na následujícím obrázku.



## Klasifikace detekovaných objektů

Pro klasifikaci detekovaných objektů je použita metoda K Nearest Neighbour (KNN) (OCR of Hand-written Data using kNN, 2014). Ovšem místo rukou psaných čísel detekujeme symboly vyskytující se v notových zápisech. Jsou to noty celé, půlové, čtvrťové, osminové a šestnáctinové, dále tyto samé pomlky, houslový klíč, oddělení taktů, křížky, béčka, odrazy a předznamenání. Všechny ostatní symboly zanedbáváme. Všechny symboly jsou normalizované pro invarianci vůči velikosti. Sada vzorů pro naučení klasifikátoru byla vytvořena ze vzorových partů, které jsme rozdělili po jednotlivých symbolech do příslušných tříd.

# Vyhodnocení

**matej**

# Závěr

**matej**

# Technologie

* python 2.7 (opencv pro Windows nepodporuje python 3.3)
* opencv verze 2.4.10 [viz.](http://www.lfd.uci.edu/%7Egohlke/pythonlibs/#opencv)
* Závislosti:
  + [Numpy](http://sourceforge.net/projects/numpy/files/NumPy/1.7.1/numpy-1.7.1-win32-superpack-python2.7.exe/download)
  + [Matplotlib](https://downloads.sourceforge.net/project/matplotlib/matplotlib/matplotlib-1.3.0/matplotlib-1.3.0.win32-py2.7.exe)
  + Matplotlib má další závislosti, které si neumí nainstalovat sám. Nám stačilo doinstalovat následující balíčky. Všechno je možné najít tady http://www.lfd.uci.edu/~gohlke/pythonlibs/
  + python-dateutil-2.2.win32-py2.7
  + pyparsing-2.0.3.win32-py2.7

# Reference

*OCR of Hand-written Data using kNN*. (23. prosinec 2014). Načteno z OpenCV 3.0.0-dev documentation: http://docs.opencv.org/trunk/doc/py\_tutorials/py\_ml/py\_knn/py\_knn\_opencv/py\_knn\_opencv.html

* [http://msw3.stanford.edu/~mmakar/mentorship/ee368\_4.pdf](http://msw3.stanford.edu/%7Emmakar/mentorship/ee368_4.pdf)
* <https://github.com/acieroid/overscore>
* [http://www.ece.rutgers.edu/~kdana/Capstone2012/Reports/CDG3.pdf](http://www.ece.rutgers.edu/%7Ekdana/Capstone2012/Reports/CDG3.pdf)