Sprawozdanie - Detekcja Nut

Sebastian Michoń 136770, Marcin Zatorski 136834

1 Wstęp

Naszym zadaniem była detekcja nut. Podzieliliśmy ten problem na dwie części. Pierwsza to wstępne przetwarzanie - binaryzacja obrazka - oraz detekcja i usunięcie pięciolinii za pomocą algorytmu grafowego. W drugiej wykrywamy nuty w oparciu o dopasowanie wzorców, a następnie klasyfikujemy je używając prostych reguł. Rozwiązanie używa OpenCV w Pythonie i jest w formie Notebooku Jupytera.

2 Preprocessing i wykrywanie pięciolinii

1. Wpierw obrazek poddany zostaje procesowi adaptatywnej binaryzacji: korzystam z gaussowego adaptatywnego thresholdingu, biorę ok. 135 sąsiednich punktów. Obrazek oryginalny jest kolejno poddany Canny edge detection i kilku kernelom jedynek rozmiaru 5x5: dzięki temu znajduję pięciolinie w formie blobów. Zaciemniam zbinaryzowany obrazek tam, gdzie nie ma kandydatów na pięciolinię

2. Następnie obraz jest rotowany. Rotacja przebiega na pierwotnym, niezbinaryzowanym obrazku, kąt później jest używany do rotowania zbinaryzowanego obrazka. Sama rotacja polega na:

   (a) Stworzeniu canny image'a pierwotnego obrazu

   (b) Znalezieniu linii na canny image'u funkcją HoughLinesP

   (c) Podziale linii ze względu na cosinusa - linie idą z lewej do prawej, jeśli xlewy==xprawy to z góry do dołu, a zatem cosinus będzie przyjmował wartości z przedziału <-1;1>, pokrywając wszystkie możliwe kąty (0;180) deg. Następnie dodaje do tablicy dp[floor(cos(a)*100)+100] +=len(a), gdzie len(a) - długość linii - dzięki temu dla każdego cosinusa kąta wiem, jaka jest suma długości linii, których cosinus kąta wynosi właśnie tyle

   (d) Znalezieniu y=acos((argmax(dp)-100)/100) (najlepiej pokryty arcus cosinusa) i obróceniu zbinaryzowanego obrazka tak, aby ten kąt stał się kątem 0 stopni w obróconym obrazku. - czyli był równoległy do góry obrazka

   (e) Powiększenie obrazka o 10 pkt z lewej, prawej, góry, dołu.

   (f) Ewentualnie później dokonuje ponownej binaryzacji - jakiś niezadokumentowany bug w opencv sprawia, że czasem po rotacji macierzą obrazka w funkcji warpAffine zmienia on swoje kolory

3. Następnie na obrazku poszukiwane są linie tworzące pięciolinię:

   (a) Robię spacer w dół obrazka: dla ustalonego sv (normalnie sv=2) robię spacer po malejącej współrzędnej y dla x=i/sv, gdzie i=(1 ... sv-1)

(b) Kończę jeśli w 7 kolejnych x-ach img[y,xl-3:xl+3] znajduję się choć 1 białe pole - przerywam spacer i zapuszczam bfs-a

(c) BFS spaceruje po wszyskich dostępnych puktach obrazka, poczynając od punktu środkowego img[y,x], jeśli może. Kończę działanie obecnej gałęzi, gdy

i. Wejdę na k-te czarne pole z rzędu (k=6)

ii. Przekroczę barierę ruchów w pionie (limes=6; gdy idę w pionie na białe pole, m+=3; na czarne: m+=4; poziomo: m=max(m-1, 0); gdy m>limes - koniec)

iii. Przekroczę obecną gałęzią BFS-a limit kolejnych ruchów, które nie ulepszają najdalszej w pionie/poziomie ścieżki kończącej się na białym (limit=50)

(d) Po zakończeniu BFS-a znajduję najdłuższą ścieżkę z lewej i z prawej - złożenie tych ścieżek daje mi ścieżkę idącą gdzieś przez obrazek

(e) Jeśli długość linii jest większa niż jakaś uzależniona od rozmiaru obrazka wartość: mówię, że to może być fragment pięciolinii. Idę po wszystkich punktach tej ścieżki, znajdując w procesie jej grubość jako średnią ilość punktów białych w górę/dół od punktu ścieżki. Następnie znowu przechodzę ścieżkę: Jeśli liczba punktów w górę/dół od punktu jest mniejsza równa sufitowi grubości, punkty te zostają skąpane w ciemności; jeśli jest wręcz przeciwnie, zakładam, że to są nuty i ich nie dotykam.

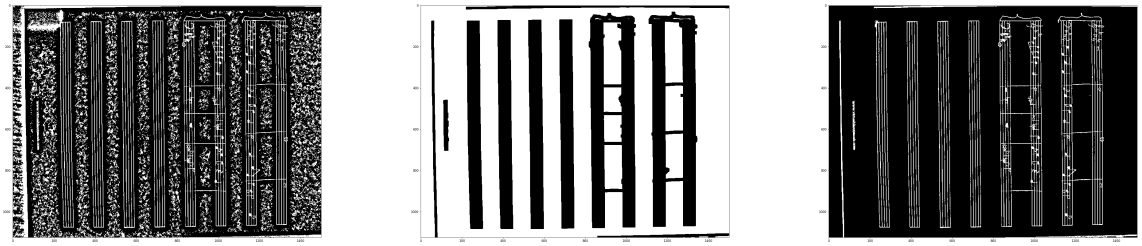
(f) Procedurę powtarzam do przejścia w pionie całego obrazka

4. Następnie poszukuję pięciolinii wśród wszystkich linii jakie znalazłem:

(a) Dla każdej linii znajduję średni punkt y-kowy, w którym ona się znajduje

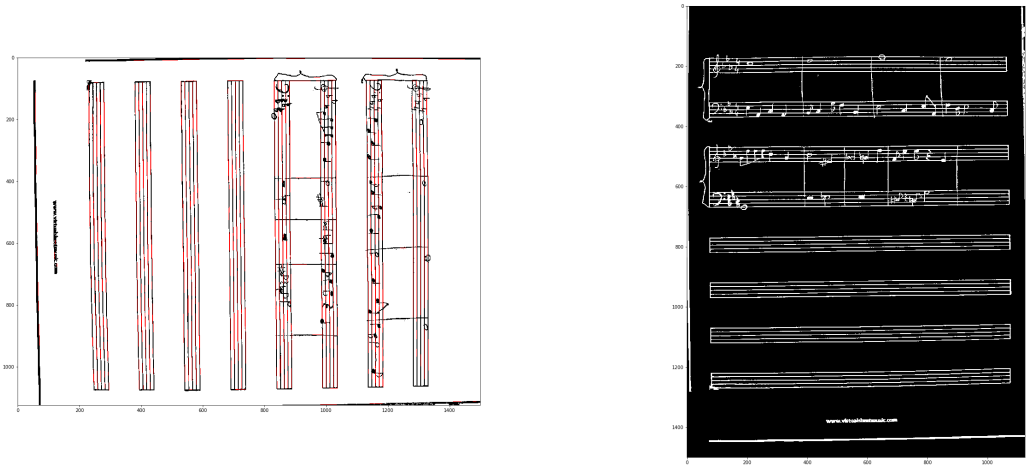
(b) Dla całego zbioru linii, zapuszczam strukturę zbiorów rozłącznych; łączenie 2 punktów sąsiednich po mean(y) zachodzi, jeśli są one odpowiednio blisko siebie

(c) Jeśli jakiś zbiór ma nie mniej niż 3 elementy, traktuje go jako pięciolinię



(a) Binaryzacja (b) Blob (c) Oczyszczona binaryzacja

Figure 1: Proces Binaryzacji.



(a) Linie proste (b) Obrót

Figure 2: Proces Rotacji.

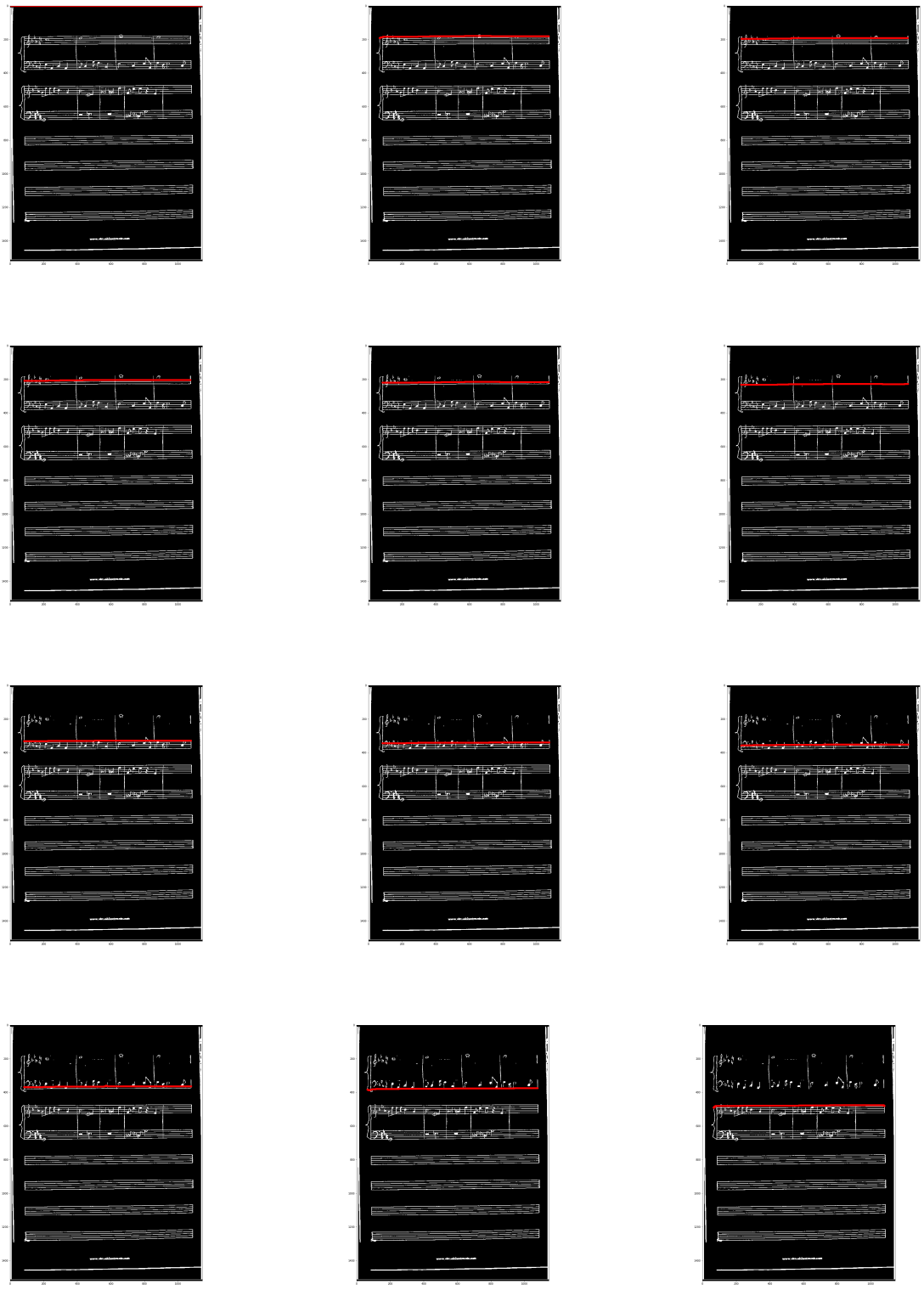


Figure 3: Proces Detekcji i usuwania linii 1.

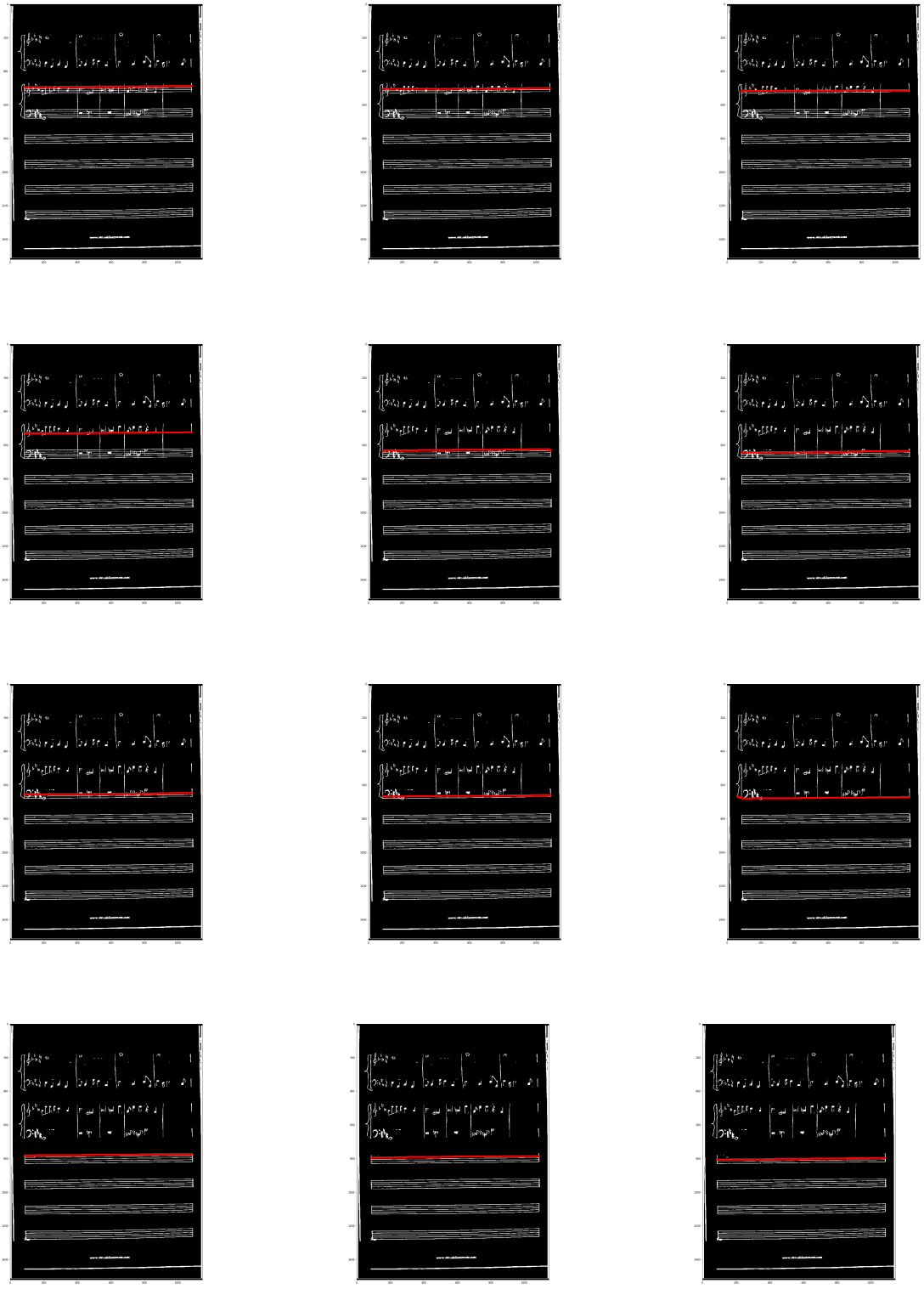
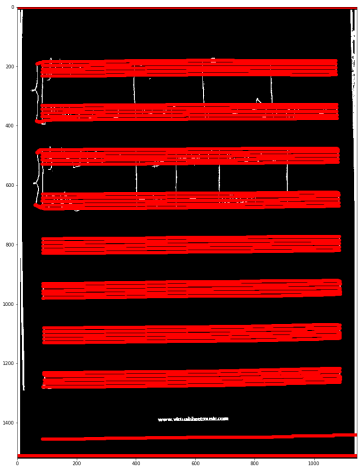
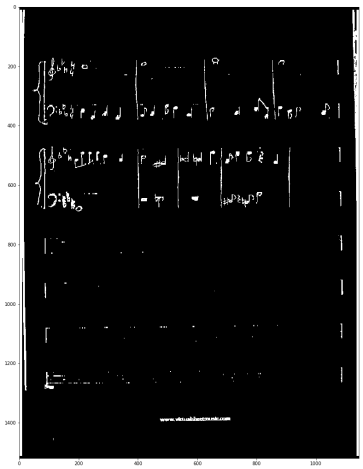


Figure 4: Proces Detekcji i usuwania linii 2.

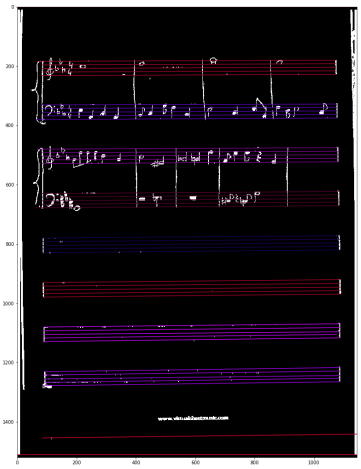


(a) To, co ścięto

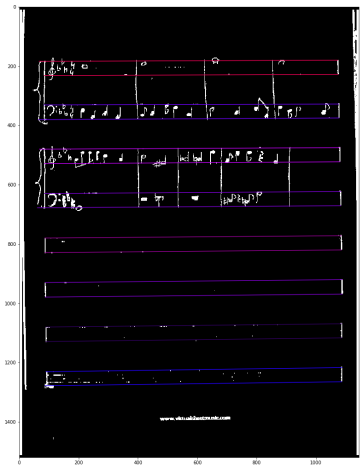


(b) To, co zostało

Figure 5: Ostateczny efekt BFS-a.



(a) Wszystkie linie zgrupowane



(b) Linie po odsiewie

Figure 6: Poszukiwanie pięciolinii

3 Detekcja nut

Kolejnym krokiem jest detekcja nut na przetworzonym obrazku.

1. Najpierw liczę rozmiary pięciolinii. Wysokość, początek i koniec są wyliczane na podstawie linii otrzymanych w poprzedniej części - biorę medianę z tych wymiarów.

2. Kolejnym krokiem jest oczyszczenie pięciolinii. Jeśli ich długość lub wysokość odbiega za bardzo od mediany to zmieniam ich rozmiary.

3. Następnie klasyfikuję klucze będące na początku pięciolinii. Robię to patrząc na ilość bieli na początku - więcej to klucz wiolinowy, mniej - klucz basowy.

4. Czwartym krokiem jest detekcja główek nut:

   (a) Najpierw zmieniam rozmiar wzorców zależnie od mediany wysokości pięciolinii.

   (b) Następnie znajduję dopasowania wzorca (używam znormalizowanej różnicy kwadratów). Do zwróconych wartości stosuję progowanie. To co poniżej progu jest nutą.

   (c) Jeśli znalazłem w małym obszarze wiele dopasowań to biorę to które ma najmniejszą wartość.

   (d) Powtarzam to dla trzech wzorców - główki wypełnionej, półnuty oraz całej nuty. Jeśli dopasowania się nakładają to zostawiam tylko pierwsze dopasowanie.

   (e) Wielkość wzorca określa prostokąt ograniczający nutę - środek prostokąta uznaję za środek nuty.

5. Teraz odfiltrowuję znalezione główki nut - jeśli są za bardzo na lewo lub na prawo od pięciolinii lub za daleko od środka najbliższej pięciolinii to je usuwam.

6. Określam wysokość nuty - najpierw znajduję pięciolinię której środek jest najbliżej nuty. Dla tej pięciolinii obliczam średnią współrzędną y dolnej i górnej linii oraz wysokość. Na podstawie odległości środka nuty od dolnej linii określam gdzie na pięciolinii nuta leży (to jest która linia lub przerwa). Znając położenie nuty oraz klucz danej pięciolinii mogę określić wysokość nuty.

7. Kolejnym etapem jest określenie długości nuty

   (a) Najpierw określam czy główka nuty jest wypełniona czy nie. Robię to patrząc na średni kolor środka nuty na oryginalnym obrazku.

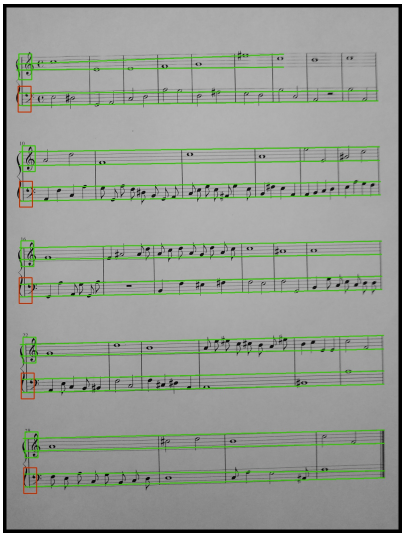
   (b) Kolejnym krokiem jest znalezienie ogonka nuty. W tym celu patrzę na projekcję pionową poniżej i powyżej nuty i określam czy jest tam odpowiednio wysokie maksimum. Jeśli jest to próbuję znaleźć koniec laseczki przechodząc po białych pikselach, pionowo, tam gdzie znalazłem maksimum. Czasem część laseczki jest usunięta - algorytm może ominąć 3 czarne piksele.

   (c) Jeśli nuta była pusta w środku i znalazłem ogonek nuty to jest to półnuta, jeśli nie to cała nuta. Jeśli główka była wypełniona i nie znalazłem ogonka to taką nutę usuwam.

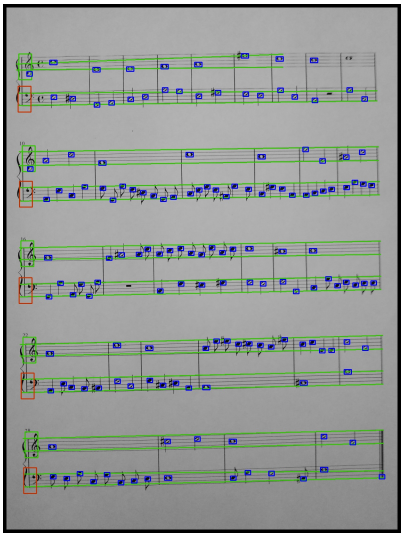
   (d) Następnie określam czy dana nuta to ósemka. Najpierw używam projekcji "po przekątnej" - średnia koloru na prawo od nuty i odpowiednio na dół lub do góry, pod kątem 45 deg. Jeśli jest tam odpowiednio wysokie maksimum to jest to flaga ósemki. Jeśli tak nie jest to sprawdzam czy na lewo lub na prawo od końca ogonka nuty jest odpowiednio dużo białego - to jest daszek ósemki.

8. Przedostatni krok to znalezienie krzyżyków. Najpierw sprawdzam dla każdej nuty pewien obszar przed nią: sprawdzam maksimum w projekcji wertykalnej i horyzontalnej - muszą być odpowiednio duże. Średni kolor też musi być odpowiednio duży. Jeśli te warunki są spełnione to sprawdzam czy obszar nie nakłada się na inną nutę - jeśli tak to odrzucam. Jeśli nie odrzucę obszaru to znajduje się w nim krzyżyk.

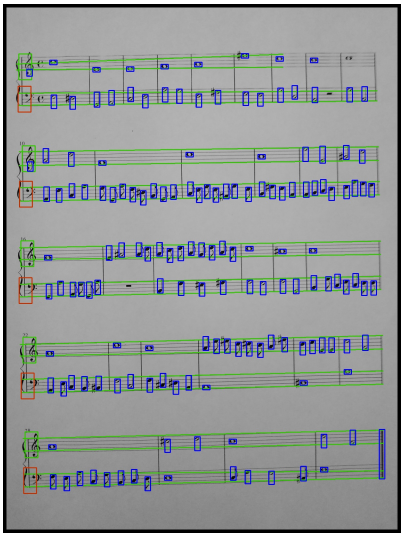
9. Na końcu wyświetlam na obrazku prostokąty ograniczające nuty. Kolory oznaczają: niebieski - cała nuta, cyan - półnuta, zielony - ćwierćnuta, magenta - ósemka. Krzyżyki są zaznaczone na żółto, klucz wiolinowy na zielono, basowy na brązowo. Nad nutami wypisuję wysokość nuty.



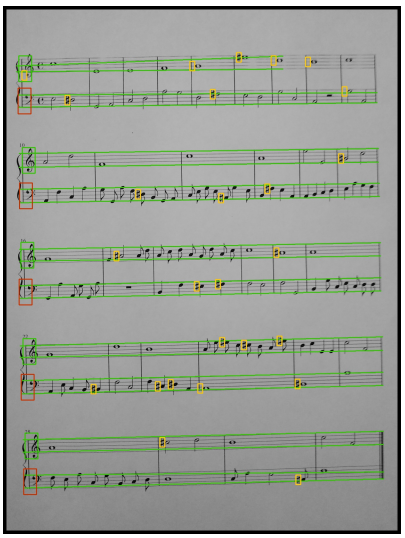
(a) Klucze i pięciolinie



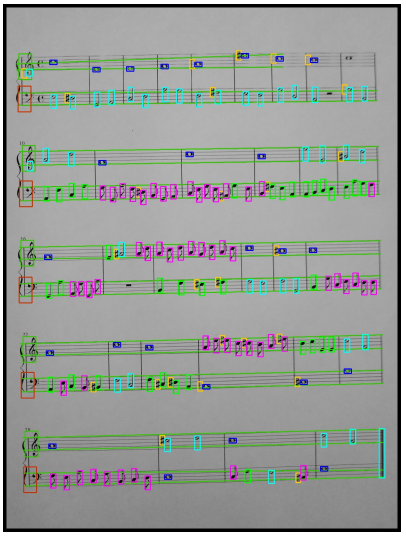
(b) Wykryte główki nut



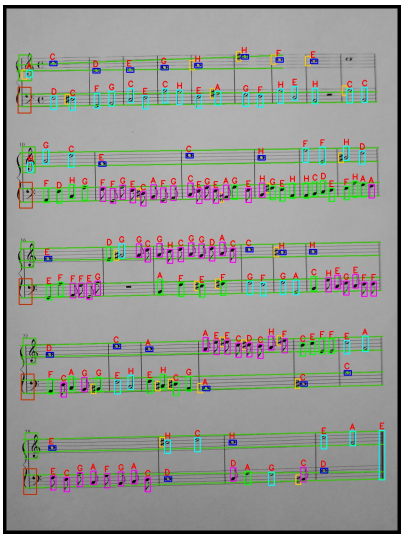
(c) Nuty z ogonkami



(d) Krzyżyki



(e) Efekt końcowy



(f) Efekt końcowy - razem z wysokością nuty

Figure 7: Etapy detekcji nut

4 Wyniki

5 Posumowanie

Detekcja nut najgorzej radzi sobie z nutami pisanymi odręcznie. Dopasowanie wzorców nie sprawdza się tu najlepiej. Nawet jeśli nuta zostanie wykryta to jest długość jest źle określana - program interpretuje nie do końca zamalowaną główkę jako otwartą. Pewnym rozwiązaniem mogło by być dodanie dodatkowych wzorców oraz zastosowanie dodatkowych przekształceń obrazu (morfologia, wyrównanie histogramu). Znajdowanie wysokości nut również nie działa idealnie jeśli linie nie są dokładnie równoległe do krawędzi obrazka lub nie są idealnie określone. Zamiast średniej można by zastosować liniową interpolację między współrzędnymi. Wykrywanie nut nie działa idealnie przy przekształceniu perspektywicznym - wzorce mają jeden rozmiar dla całej pięciolinii. Można by spróbować podzielić kartkę na pojedyncze pięciolinie i stosować wzorce na tych pięcioliniach. Problematyczne są także przypadki gdy kartka zostanie obrócona tak, że klucze są po prawej stronie - przy rotacji nie jest uwzględnione położenie kluczy.