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, detekcja i usunięcie pięciolinii, a druga to detekcja i klasyfikacja nut.

2 Technika wykonania algorytmu

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 zarotowaniu zbinaryzowanego obrazka w tak, aby ten kąt stał się kątem 0 stopni w zarotowanym 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 gałęznią 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 góre/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.

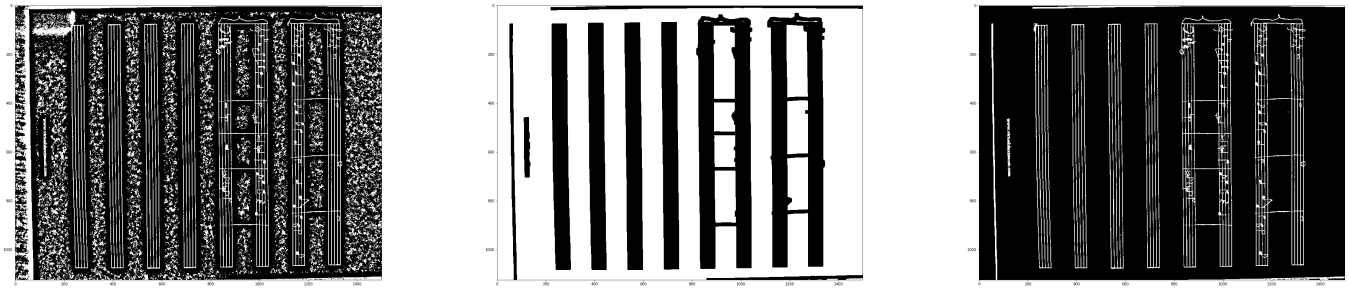
(f) Procedurę powtarzam do przejścia w pionie całego obrazka

4. Następnie poszukuję pięciolinii wśród wszystkich linii jakie znalazłem:

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

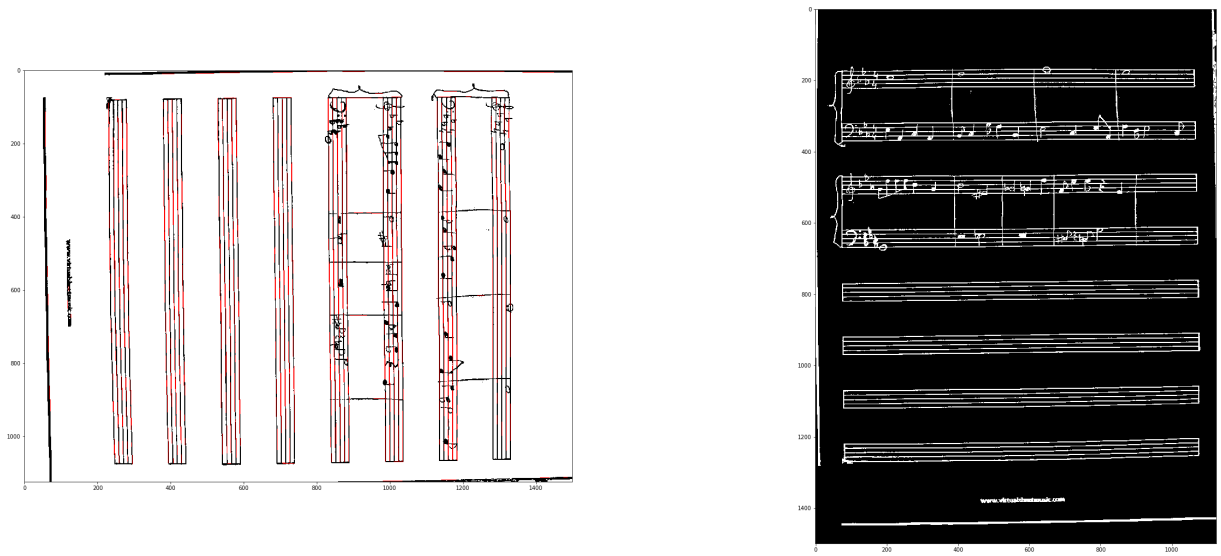
(b) Dla całego zbioru linii, zapuszczam strukturę zbiorów rozłącznych; łączenie 2 punktów sąsiednich po mean(y) zachodzi, jeśli są one odpowiednio blisko siebie

(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.

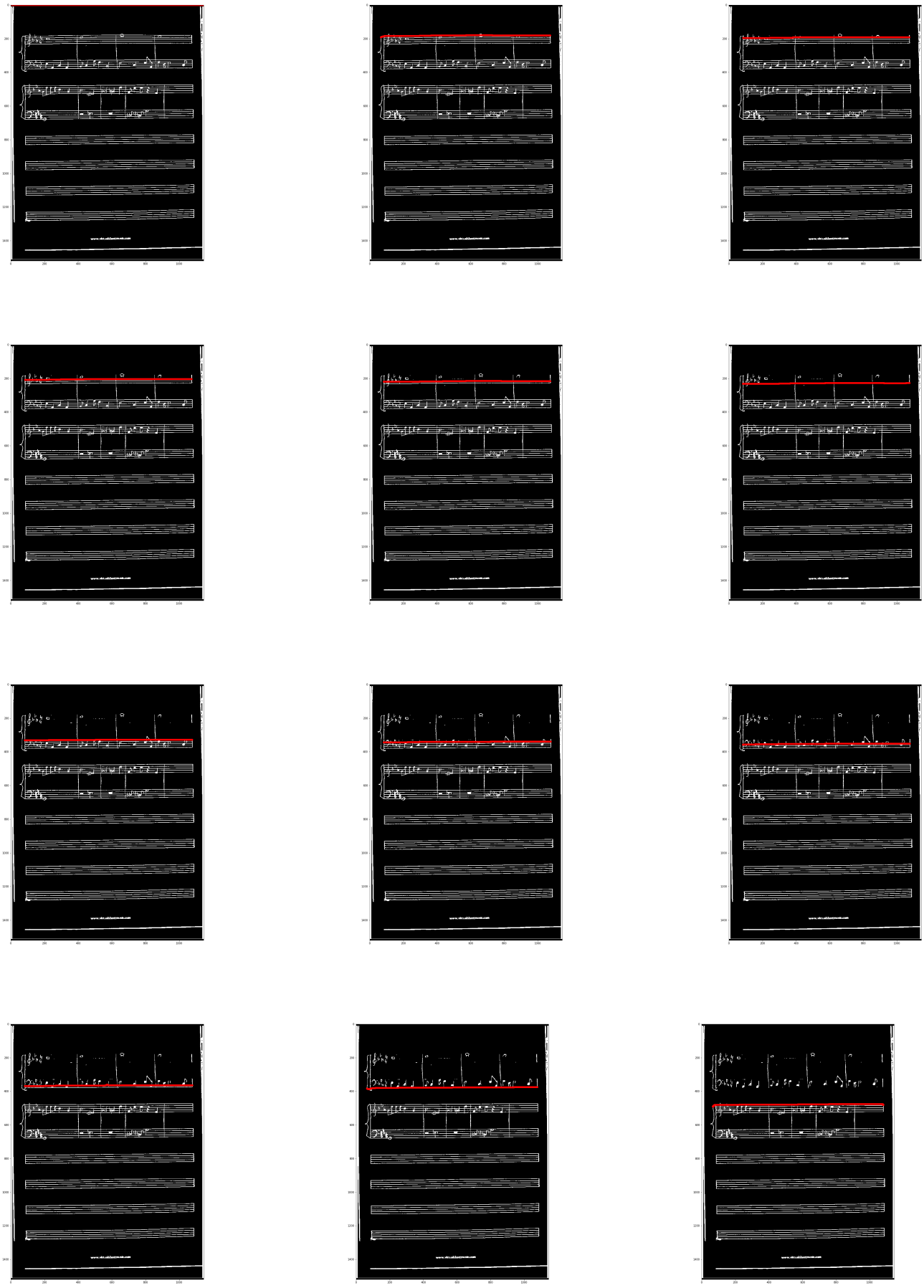


Figure 3: Proces Detekcji i usuwania linii 1.

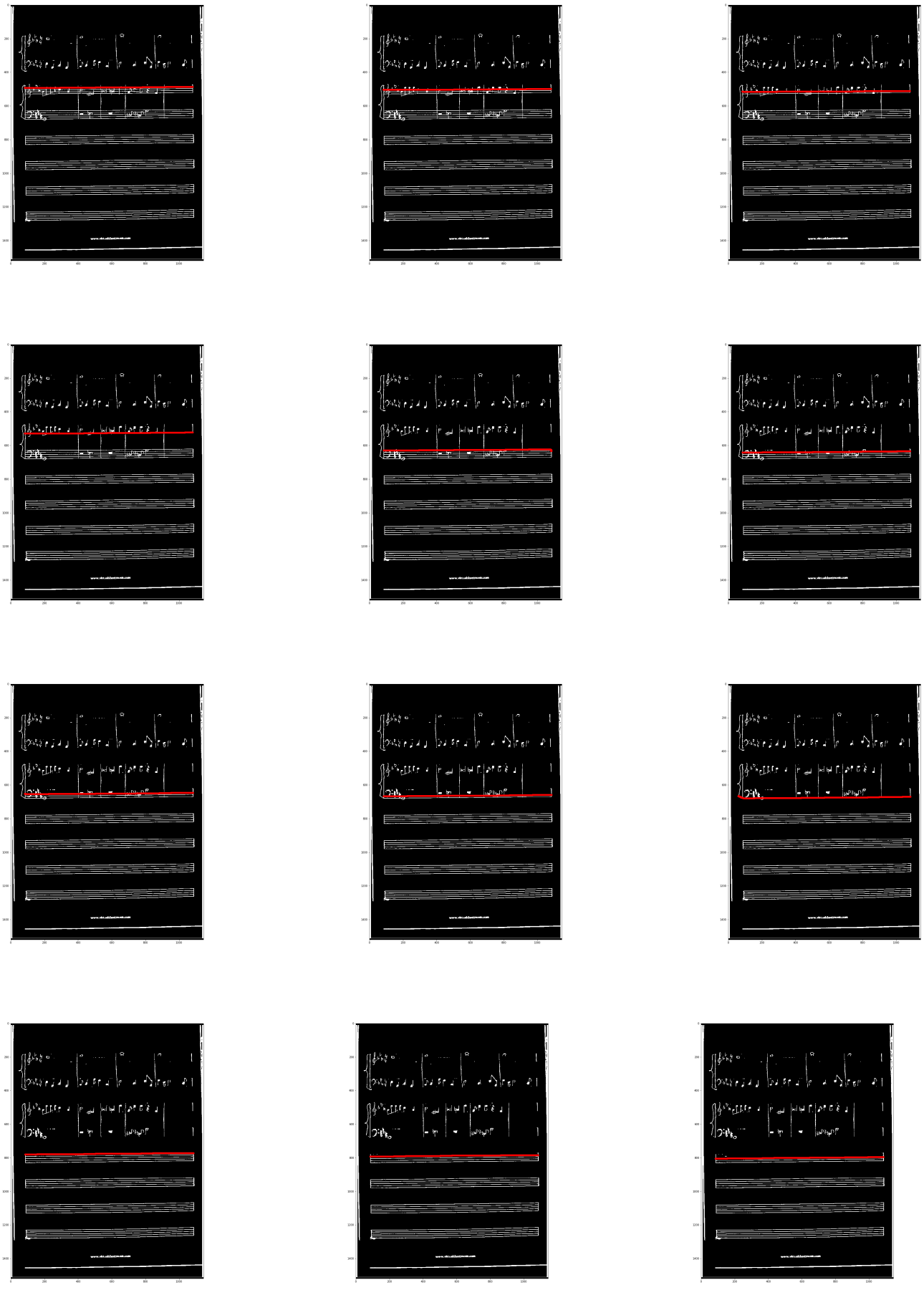
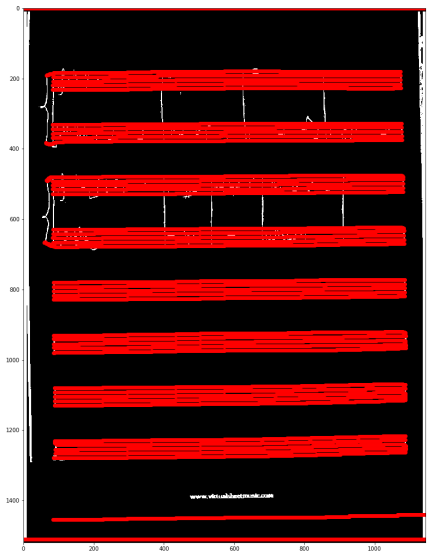
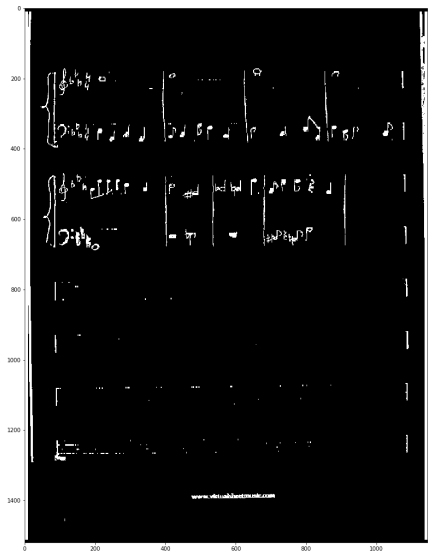


Figure 4: Proces Detekcji i usuwania linii 2.

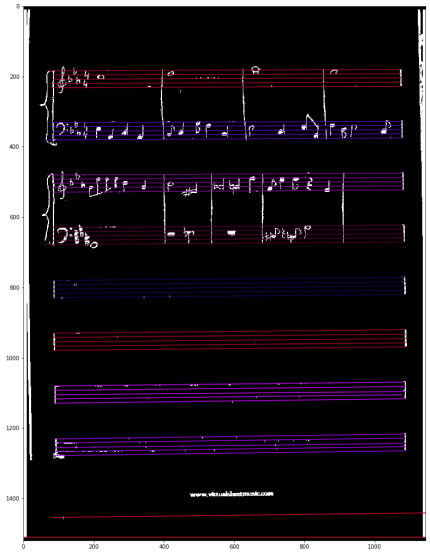


(a) To, co ścięto

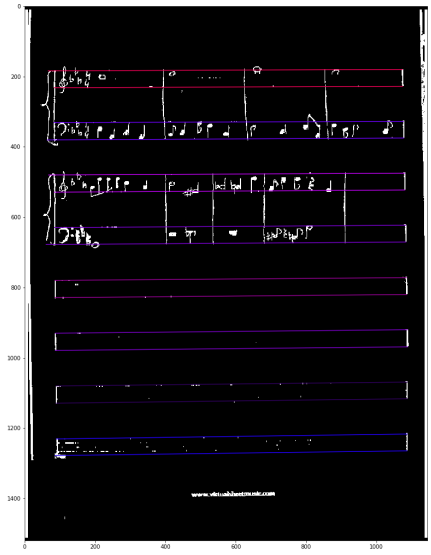


(b) To, co zostało

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 zwróconych w poprzednim kroku - 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. Trzecim krokiem jest detekcja główek nut:

   (a) Najpierw zmieniam rozmiar wzorców zależnie od 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.

5. Teraz odfiltrowuję znalezione główki nut - jeśli są zbyt daleko od końców pięciolinii to je usuwam.

6. Znajduję wysokość nuty - najpierw obliczam gdzie na pięciolinii nuta leży. Biorę środek nuty - obliczony jako środek geometryczny wzorca + lewy górny róg. Porównuję to do współrzędnej y dolnej linii - obliczonej jako średnia y początku i końca. Na podstawie obliczonego położenia nuty i klucza danej pięciolinii określam 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 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 skosie" - na prawo od nuty i odpowiednio na dół lub do góry. 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. Ostatni krok to narysowanie prostokątów otaczających nuty oraz naniesienie informacji o wysokości nuty.