

Komunikacja Człowiek-Komputer

rozpoznawanie nut

kubawasik

November 2018

1 Zastosowanie

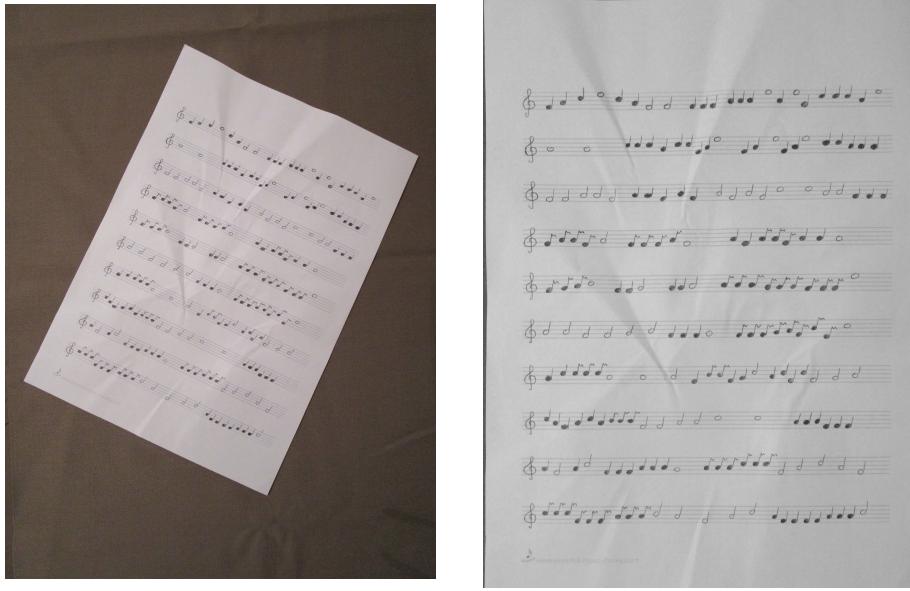
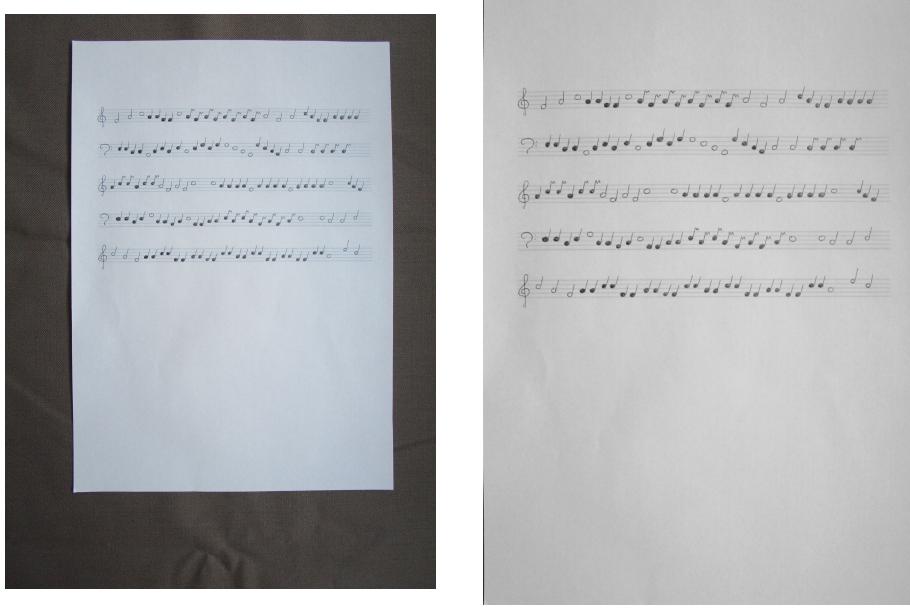
Program służy do wykrywania nut ze zdjęcia. Jest w stanie odnaleźć klucze (wiolinowy i basowy) oraz wypisać kolejność pojawienia się nuty, jej wysokość oraz numer pięciolini, w której się znajduje.

2 Przetwarzanie

Opiszemy najważniejsze kroki, jakie podjęliśmy, aby wykryć nuty.

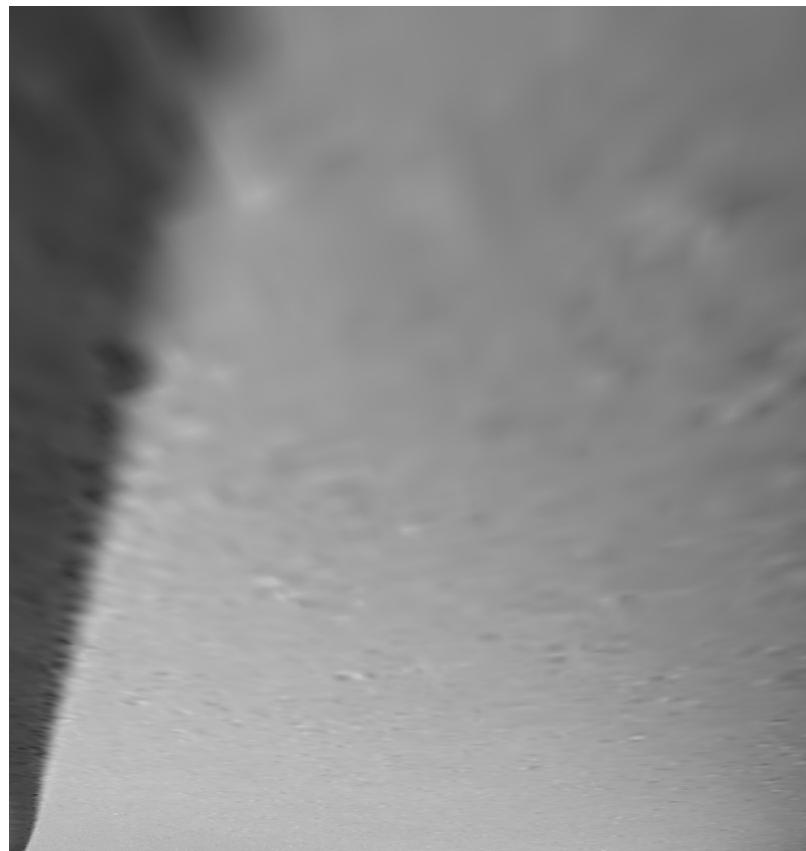
2.1 Wycinanie strony

Pierwszym krokiem, który należało wykonać było "wyłuskanie" zdjęcia samej kartki. W tym celu poszukiwaliśmy największego konturu na obrazie, który ma cztery narożniki, zakładając, że to nasza kartka. Aby było to możliwe, wcześniej nakładaliśmy filtr Canny. Kiedy już odnaleźliśmy nasze narożniki, po odpowiednim ich posortowaniu używaliśmy dla nich funkcji `cv2.getPerspectiveTransform()` oraz `cv2.warpPerspective()`, aby odpowiednio znaleźć macierz transformacji i zastosować ją na zdjęciu.



Rysunek 2: Obraz oryginalny i po wycięciu

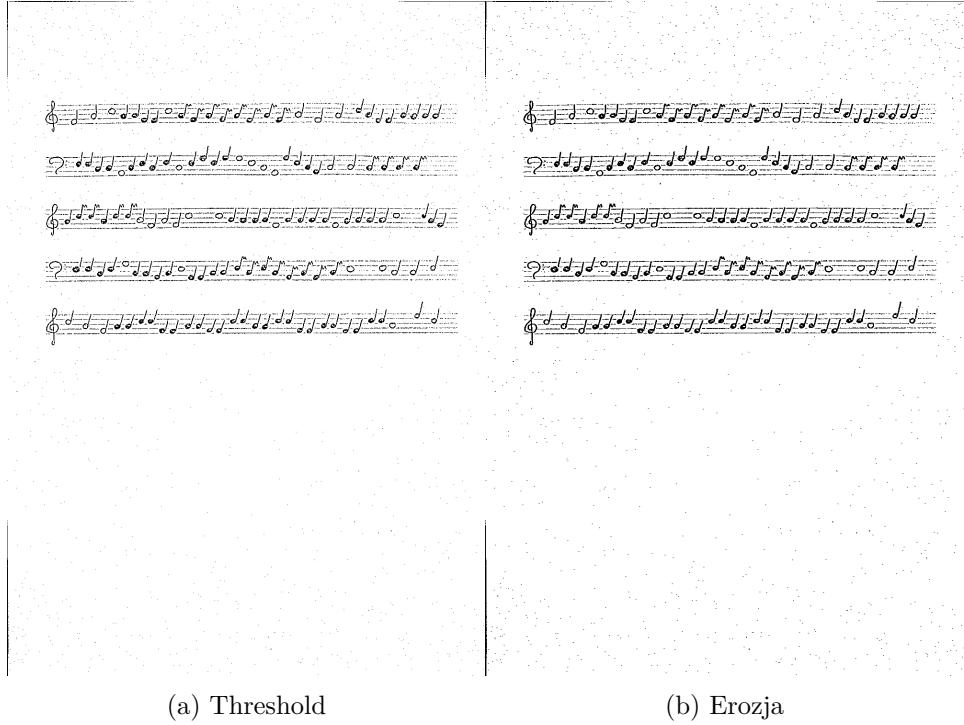
Nie dla wszystkich przypadków wykrywanie kartki zadziałało - dla nierównomiernego oświetlenia wykryte kontury kartki nie były zamknięte. Efekt takiego wycięcia wyglądał tak:

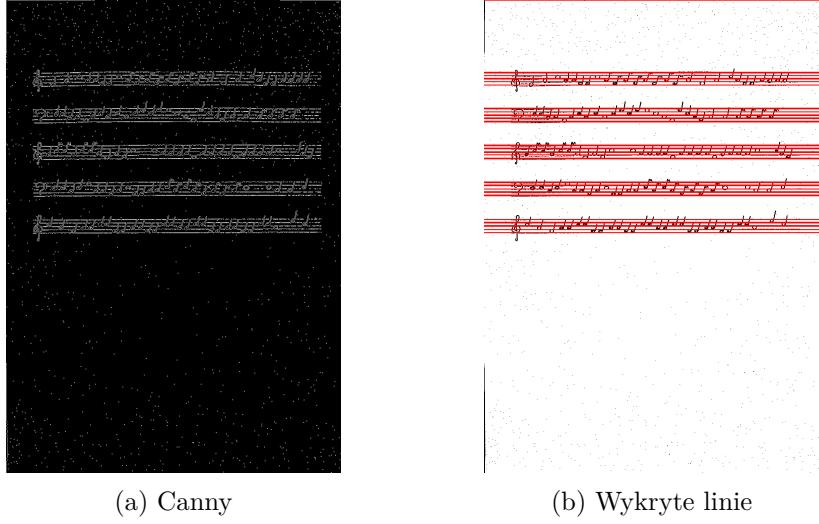


Rysunek 3: Nieudany przykład

2.2 Wykrywanie pięciolinii

Kolejnym zadaniem było wykrycie wysokości, na których znajdują się pięciolinie. Zastosowaliśmy do tego transformacje Hougha do wykrywania linii. Skorzystaliśmy z gotowej funkcji **cv2.HoughLines**. Jako jej parametry podaliśmy $\rho = 1$ oraz $\theta = \pi/100$ (ponieważ poszukujemy linii poziomych, kąt θ będzie niewielki, tak samo jak długość ρ). Wcześniej jednak trzeba było przetworzyć obraz za pomocą funkcji `threshold_local` oraz nakładając filtr Canny. Funkcja ta w zależności od jasności obrazu lepiej działała z parametrem 'method' ustawionym na 'median' albo 'mean'. Zbyt duży `threshold` usuwał nutki, zbyt mały pozostawiał szумy i utrudniał wykrywanie nut. Ostatecznie wybraliśmy nieco mniejszy i dodaliśmy erozję (`cv2.erode()`).

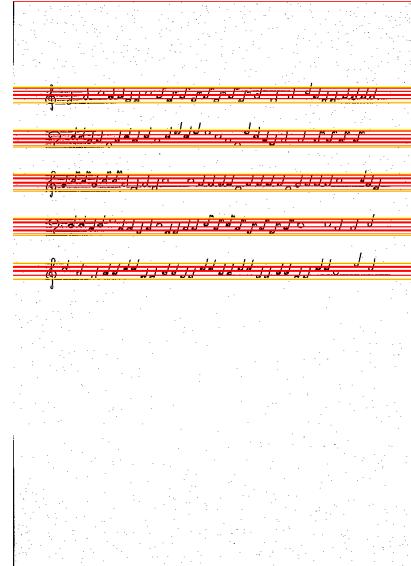




(a) Canny

(b) Wykryte linie

Po wykryciu linii za pomocą transformacji Hougha obliczamy współrzędne jej końców, a następnie centra. Sprawdzając odległość pomiędzy liniami, stwierdzamy, czy należą one do jednej pięciolinii, czy też do dwóch różnych. Jeśli znajdziemy pięć równoległych, leżących blisko siebie linii, zakładamy, że krańcowe z nich są granicami jednej z pięciolinii. Zwracamy obiekt Staff, który zawiera granice pięciolinii.

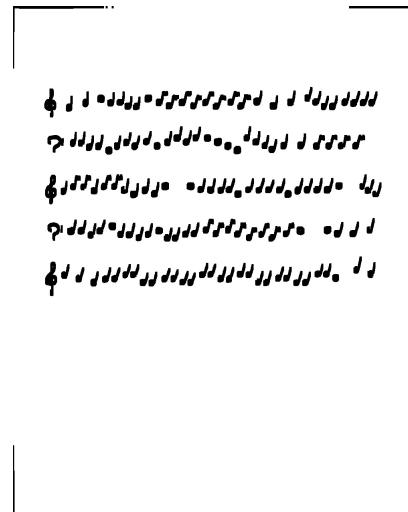


Rysunek 6: Wykryte pięciolinie

2.3 Usuwanie pięciolinii

Kolejnym krokiem jest usunięcie pięciolinii w celu łatwego wykrycia konturów nut. W tym celu stosujemy na obrazie threshold, a następnie obracamy go w negatyw. Do usunięcia pięciolinii używamy funkcji

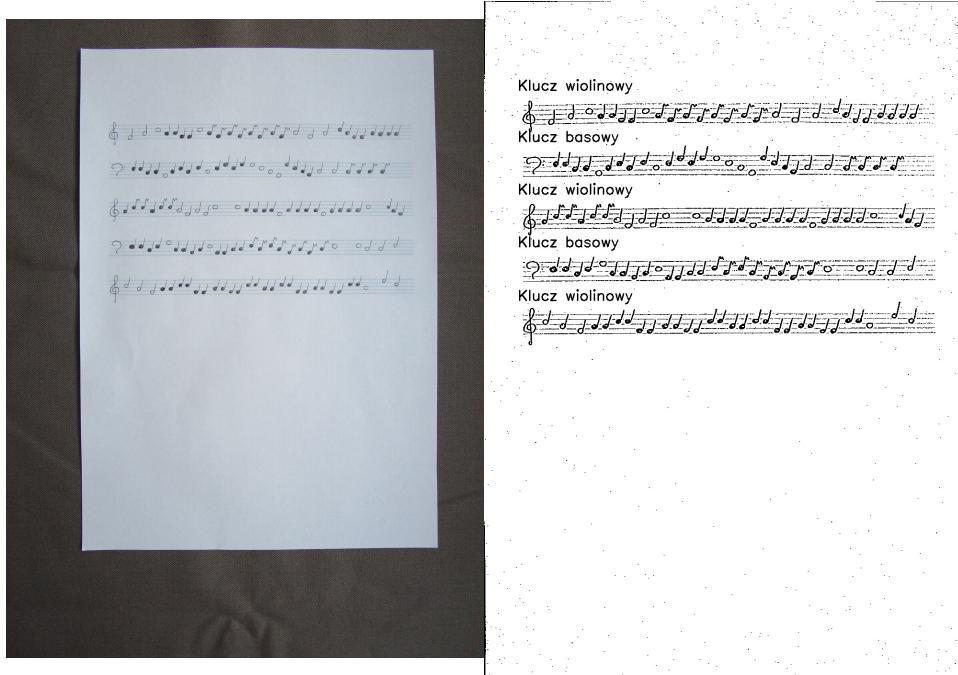
cv2.getStructuringElement oraz **cv2.morphologyEx**, z których pierwsza tworzy element, który będzie wyszukiwany i usuwany (w naszym przypadku będzie to długi cieński prostokąt), a druga wykonuje wskazane przekształcenie morfologiczne (z parametrem **cv2.MORPH_OPEN** wykonuje otwarcie - na zasadzie dylatacji) Kolejnym krokiem jest wykonanie dość silnej erozji, aby struktura nut była pełna. Następnie wyszukujemy kontury. W przypadku wyszukiwania nut okrajamy nieco kartkę (przypisuję barwę białą pewnym marginesom), aby uniknąć wyszukania tekstu na dole strony i kluczy wiolinowych/basowych. Następnie kontury sortujemy po obwodach. Wykrycie nut całych i ósemek jest trywialne (przedziały odpowiednio najmniejszych i największych obwodów). W przypadku ćwierćnuta i półnuta, które mają ten sam rozmiar, sprawdzam średnią wartość fragmentu obrazu wydzielonego przez kontur nuty (półnuta jest biała w środku - średnia wartość znacznie wyższa).

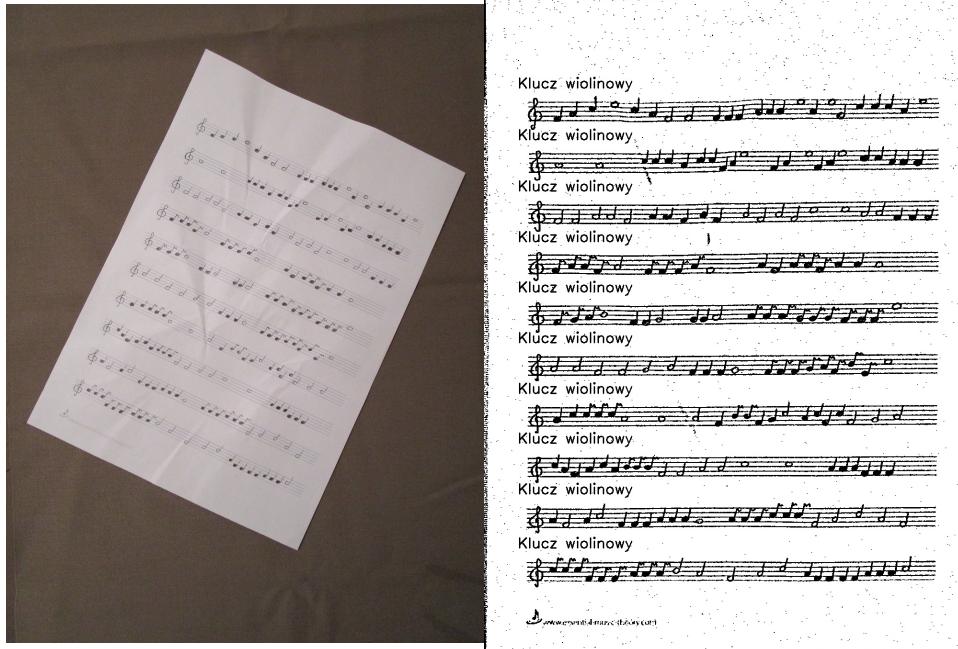


Rysunek 7: Usunięte pięciolinie + silna erozja

2.4 Wykrywanie kluczy

Po skorzystaniu z utworzonych już funkcji do przekształcenia obrazu, znajdowania położenia każdej pięciolinii oraz usuwania ich z obrazu pozostawiając możemy rozpocząć samo znajdowanie kluczy. Przy szukaniu kluczy korzystamy z wiedzy, że muszą się one zawsze znajdować na początku pięciolinii. Wyznaczmy więc potencjalne miejsce klucza, proporcjonalnie do jego rozmiarów. Następnie, korzystając z różnic pomiędzy kluczem wiolinowym, a kluczem basowym, będziemy przeszukiwać obszar pod pięciolinią. Szukamy pierwszego wystąpienia czarnego piksela. Jeśli jakiś się pojawił to znaczy że znaleźliśmy klucz wiolinowy, w przeciwnym wypadku znaleźliśmy klucz basowy.





Rysunek 9: Obraz oryginalny i i ze znalezionymi

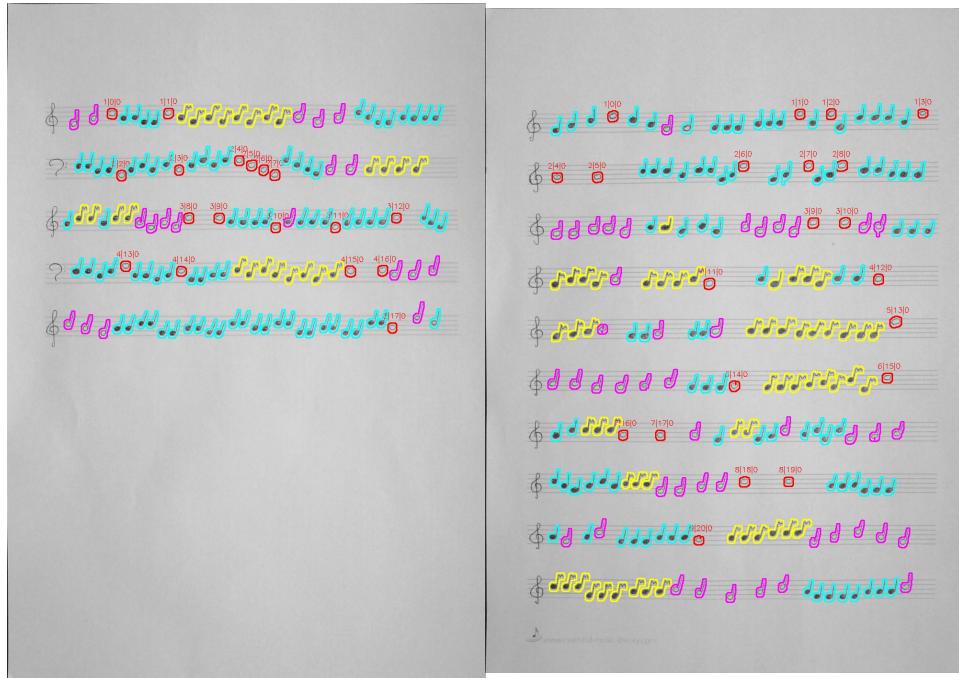
2.5 TO DO

Odpowiednie podpisywanie nut

Połączenie wykrywania klucza z wykrywaniem nut

Uszczególnienie parametrów podziału nut na całe/półnuty/ćwierćnuty/ósemki

3 Tymczasowe wyniki



The image shows two staves of musical notation. The notation consists of colored dots (red, blue, green, yellow) connected by stems, indicating pitch and rhythm. The left staff begins with a tempo of 110 BPM. The right staff begins with a tempo of 100 BPM. Both staves include a URL watermark at the bottom: www.elementalmusictheory.com.

