POLITECHNIKA WARSZAWSKA

Widzenie maszynowe

PROJEKT

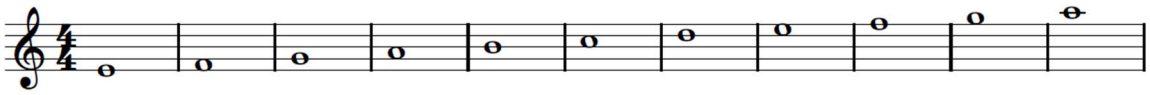
Marcin Przestrzelski
Robotyka
313876

2022/2023

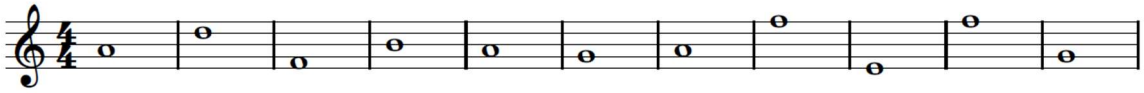
Temat:

Rozpoznawanie dźwięku nut, na pięciolinii.

Badane obrazy:



Obrazy „nuty1.PNG” i „nuty3.PNG”



Obraz „nuty2.PNG”

Do testowania używałem dwóch pięciolinii na których są wypisane kolejno nuty:
E F G A H C D E F G A na obrazach „nuty1.PNG” i „nuty3.PNG”
A D F H A G A F E F G na obrazie „nuty2.PNG”
Konieczne do testowania programu było sprawdzenie, jak rodzi on sobie z obrazami w większym formacie. Dlatego badamy też obraz „nuty3.PNG”, który jest rozszerzoną wersją obrazu „nuty1.PNG”.

Opis kodu:

Pobranie zdjęcia i jego wymiarów i zamiana go na obraz czarno-biały.

```
import cv2
import numpy as np

file_path=input("Podaj sciezke do pliku z nutami")

img = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)

cv2.imshow("Input", img)

_, img_thresh = cv2.threshold(img, 100, 255, cv2.THRESH_BINARY)

img_height=img.shape[0]
img_width=img.shape[1]
```

Wyodrębnienie znaków z obrazu:

Usunięcie linii poziomych pięciolinii, przy pomocy filtru gradientowego poziomego.

```
#detecting horizontal lines
horizontal=img_thresh.copy()

kernel_horizontal=np.ones((1,int(10*img_width/1294)))

horizontal=cv2.dilate(horizontal, kernel_horizontal, iterations=1)

horizontal=img_thresh-horizontal+255
```

Usunięcie linii pionowych pięciolinii, rozdzielających takty, przy pomocy filtru gradientowego pionowego.

```
#detecting vertical lines
vertical=img_thresh.copy()

kernel_vertical=np.ones((int(10*img_height/160),1))

vertical=cv2.dilate(vertical, kernel_vertical, iterations=1)

vertical=img_thresh-vertical+255

img_only_notes = vertical+horizontal
```

Operacja otwarcia dokonana na nutach, aby wypełniły wnętrze swojej objętości. Pozwoli nam to w następnych etapach na wyznaczenie powierzchni zajmowanej przez nutę.

```
#close notes
img_only_notes = cv2.erode(img_only_notes, np.ones((int(11*img_width/1294), int(11*img_height/160)), np.uint8))
img_only_notes = cv2.dilate(img_only_notes, np.ones((int(11*img_width/1294), int(11*img_height/160)), np.uint8))

cv2.imshow("elements", img_only_notes)
```

Wyodrębnienie konturów figur na obrazie. Dodatkowo w tym korku odrzucamy pozostałości zbędnych obiektów z filtrowania. Kryterium doboru elementów będących nutami, kluczem wiolinowym lub metrum, jest na podstawie zajmowanej powierzchni. Elementy za małe odrzucamy.

```
#get contours
img_only_notes_inverted = img_only_notes.copy()

conts, hier = cv2.findContours(img_only_notes, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
good_conts = []

for cont in conts:
    area = cv2.contourArea(cont)
    if area > 50*((img_height*img_width)/(1294*160)) and area < 100000*((img_height*img_width)/(1294*160)):
        good_conts.append(cont)
```

Analiza nut:

Wyznaczenie środka masy, każdego z elementów i położenia każdego z elementów.

```
output = img.copy()

conts_parameters = []

for cont in good_conts:
    area = cv2.contourArea(cont)
    M = cv2.moments(cont)
    cX = int(M["m10"]/M["m00"])
    cY = int(M["m01"]/M["m00"])
    note_type='c'
    conts_parameters.append([cont, note_type, cX, cY])
```

Posortowanie elementów od tego najbardziej po lewej stronie do tego najbardziej po prawej stronie.

```
#sort notes from left to right
for i in range(len(conts_parameters)):
    for j in range(0, len(conts_parameters)-i-1):
        if conts_parameters[j][2]>conts_parameters[j+1][2]:
            temp = conts_parameters[j]
            conts_parameters[j]=conts_parameters[j+1]
            conts_parameters[j+1] = temp
```

Wyznaczenie rozmiarów klucza wiolinowego, który będzie naszym odnośnikiem względem pozycji nut. Dodatkowo tworzymy współczynnik k, który mówi nam o skali obrazu względem obrazu „nuty1.PNG”.

```
treble_max=conts_parameters[0][0][1][0][1]
treble_min=conts_parameters[0][0][1][0][1]

for i in range(len(conts_parameters[0][0])):
    if(conts_parameters[0][0][i][0][1]>=treble_max):
        treble_max=conts_parameters[0][0][i][0][1]
    if(conts_parameters[0][0][i][0][1]<=treble_min):
        treble_min=conts_parameters[0][0][i][0][1]

treble_height=treble_max-treble_min

k=treble_height/89.0 #skala podbienstwa klucza wiolinowego obrazu, do wzorca
```

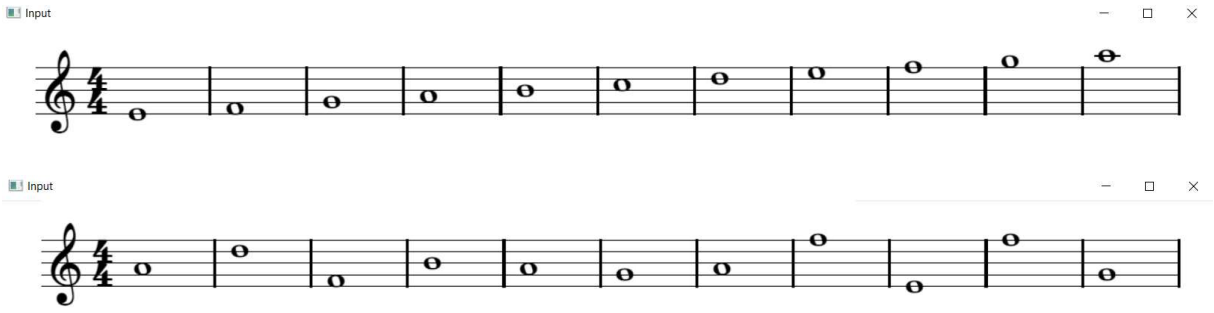
Wyznaczenie dźwięku nuty na podstawie względnej wysokości nuty, w porównaniu do klucza wiolinowego.

```
for cont in conts_parameters:
    dY=conts_parameters[0][3]-cont[3]
    if((dY>k*-31 and dY<k*-29)or(dY>k*12 and dY<k*14)):
        note_type='c'
    elif((dY>k*-25 and dY<k*-23)or(dY>k*19 and dY<k*21)):
        note_type='d'
    elif((dY>k*-19 and dY<k*-17)or(dY>k*25 and dY<k*27)):
        note_type='e'
    elif((dY>k*-13 and dY<k*-11)or(dY>k*31 and dY<k*33)):
        note_type='f'
    elif((dY>k*-6 and dY<k*-4)or(dY>k*37 and dY<k*39)):
        note_type='g'
    elif((dY>k*0 and dY<k*2)or(dY>k*43 and dY<k*45)):
        note_type='a'
    elif((dY>k*6 and dY<k*8)or(dY>k*49 and dY<k*51)):
        note_type='h'
    cont[1]=note_type
```

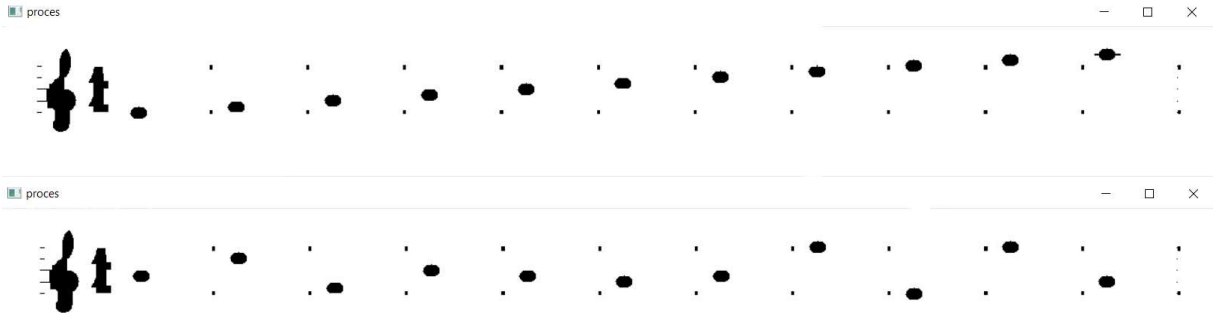
Podpisanie nut na obrazie wyjściowymi i wypisanie kolejnych nut w terminalu.

```
#add note type to output
for i in range(len(conts_parameters)):
    if (i>1):
        output = cv2.putText(output, conts_parameters[i][1], (conts_parameters[i][2]-10, int(135*img_height/160)), cv2.FONT_HERSHEY_SIMPLEX,0.8,(0,0,255),2)
notes = []
for i in range(len(conts_parameters)):
    if (i>1):
        notes.append
        print(conts_parameters[i][1], end=' ')
print()

cv2.imshow("Output", output)

cv2.waitKey()
cv2.destroyAllWindows()
```

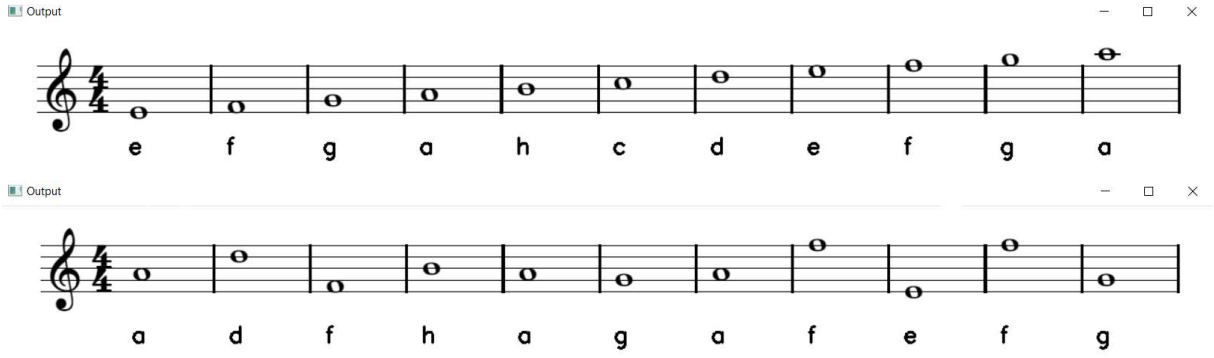
Wyniki przetwarzania:



Obrazy wejściowy.



Obrazy z wyodrębnionymi elementami.



Obrazy z podpisanymi nutami.

Wyniki w terminalu

```
= RESTART: C:\Users\Marcin Nitro5\Desktop\WMA\projekt\Marcin_Przestrzelski_proje
kt_WMA\Projekt_Marcin_PrzestrzelskiV1.py
Podaj sciezke do pliku z nutaminuty1.png
e f g a h c d e f g a
```

```
= RESTART: C:\Users\Marcin Nitro5\Desktop\WMA\projekt\Marcin_Przestrzelski_proje
kt_WMA\Projekt_Marcin_PrzestrzelskiV1.py
Podaj sciezke do pliku z nutaminuty2.png
a d f h a g a f e f g
```