

OCR

(Optical Character Recognition)

1. Opis działania programu

Działanie programu zostało oparte o algorytm znormalizowanej korelacji wzajemnej (krzyżowej) (ang. normalized cross correlation). Jego działanie polega na znalezieniu pozycji danego wzorca na dwuwymiarowej macierzy pikseli tworzącej obraz f . Niech $f(x,y)$ oznacza intensywność wartości obrazu f o rozmiarach w $M_x \times M_y$ punkcie (x,y) , gdzie $x \in \{0, \dots, M_x - 1\}$, $y \in \{0, \dots, M_y - 1\}$. Wzorzec $t(x,y)$ jest reprezentowany przez macierz o wymiarach $N_x \times N_y$. Częstym sposobem do wyliczenia pozycji wzorca t na obrazie f jest przeprowadzenie korelacji NCC, stosując wzór:

$$\frac{1}{n} \sum_{x,y} \frac{(f(x,y) - \bar{f})(t(x,y) - \bar{t})}{\sigma_f \sigma_t}, \text{ gdzie:}$$

n to liczba pikseli, \bar{f} to średnia z f , a σ_f to odchylenie standardowe.

W praktyce jednak szukany wzorzec rozbija się na sumę funkcji składowych i liczy się korelację dla każdej składowej, a następnie wyznacza sumę ważoną z nich jako wyniku, korzystając z następujących wektorów:

$$\begin{aligned} F(x,y) &= f(x,y) - \bar{f} \\ T(x,y) &= t(x,y) - \bar{t} \end{aligned} \quad \left\langle \frac{F}{\|F\|}, \frac{T}{\|T\|} \right\rangle$$

A ww. suma jest równa:

Obliczanie podobieństwa wzorca do obrazka oparte jest na zastosowaniu miary odległości euklidesowej:

$$d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$$

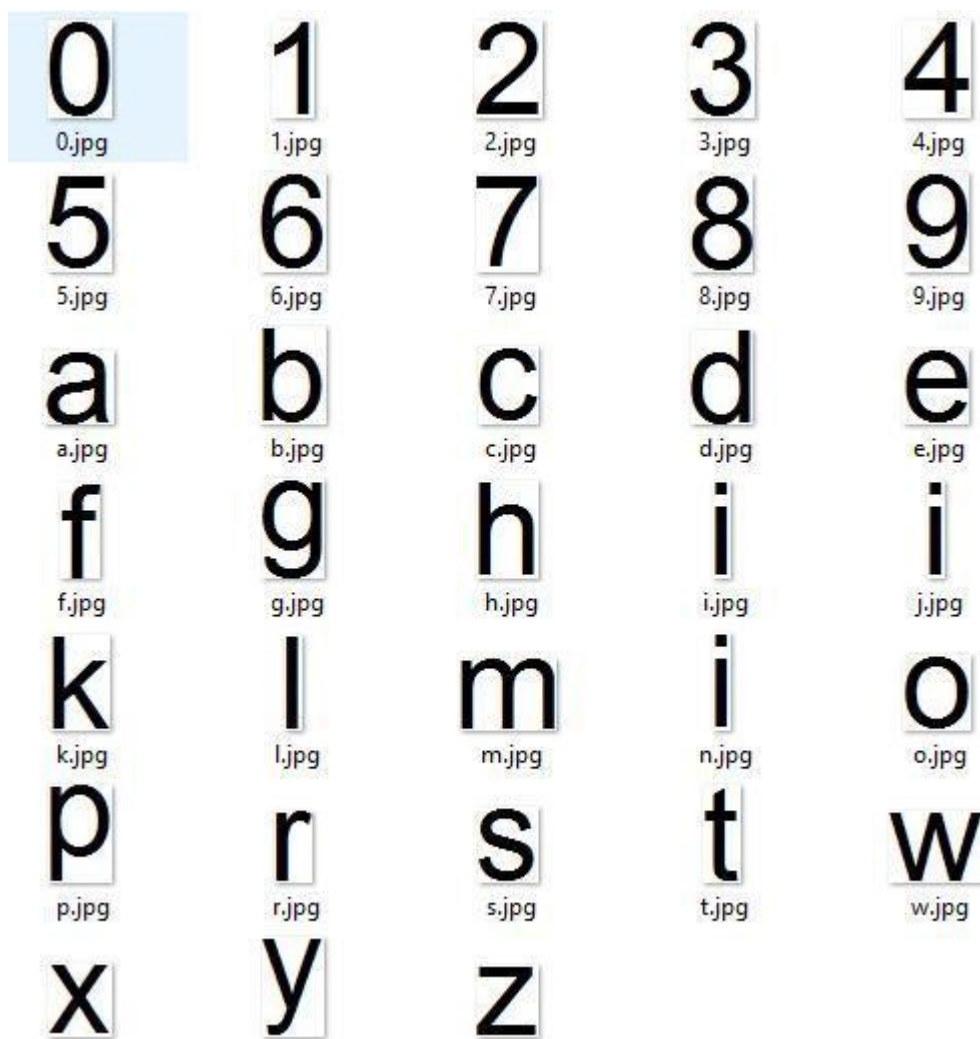
Program przyjmuje na wejściu obraz, który ma zostać przeanalizowany, oraz czcionkę, która znajduje się na obrazie w postaci wzorów każdej z występujących liter. Możemy jednocześnie dodać kilka czcionek i obrazów do przeanalizowania. Należy je tylko wkleić do odpowiedniego miejsca w projekcie. Argumenty programu można ustawić w metodzie main klasy OCRTest.

Na potrzeby programu przygotowałem wzory dwóch czcionek, jednej szeryfowej i jednej bezszeryfowej. Niestety mój program nie obsługuje obracania tekstu. W programie wykorzystałem bibliotekę axet.

2. Przykłady działania programu

a) Dla czcionek bezszeryfowych:

Wykorzystałem czcionkę Arial, w rozmiarze 48px i stworzyłem dla każdej z liter obrazek JPG, czyli nasz wzorzec.



Obraz po którym przeszukiwałem:

W wyniku otrzymałem:

this is example text
it shows how that
program works
today is 2016 year

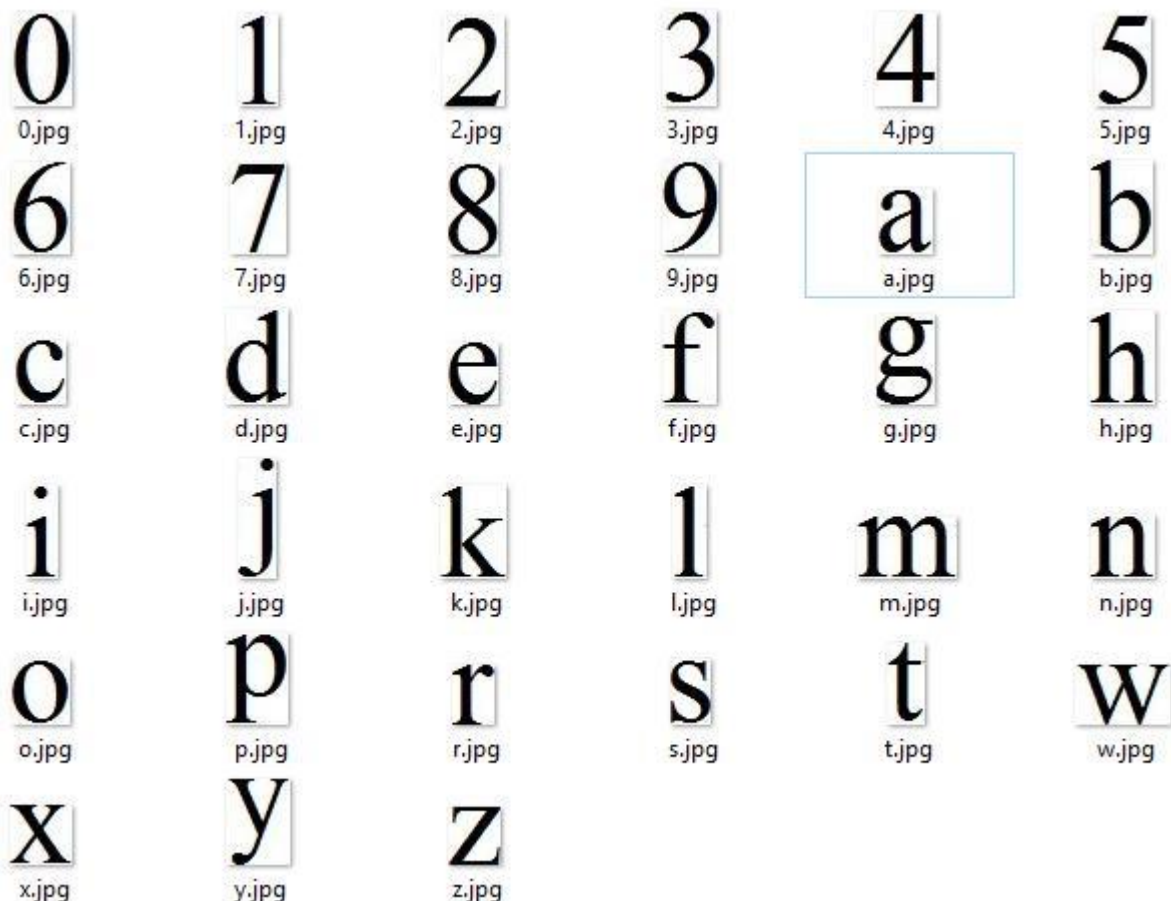
t h j s l s e x a m b j e t e x b
l t s h o w s h o w t h a t
b r o g r a m w o r k s
t o d a y l s 2 0 1 6 y e a h

I poniżej dołączam cały output jaki otrzymałem w eclipse, łącznie z ilością wystąpień każdego znaku:

```
Problems Javadoc Declaration Console
<terminated> OCRTest [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (16 cze 2016, 18:55:10)
{0=1, 1=4, 2=1, 3=0, 4=0, 5=0, 6=1, 7=0, 8=0, 9=0, a=5, b=3, c=0, d=1, e=4, f=0, g=1, h=5, i=0, j=2, k=1, l=0,
m=2, n=0, o=5, p=0, r=3, s=6, t=6, w=3, x=2, y=2, z=0}
t h j s l s e x a m b j e t e x b
l t s h o w s h o w t h a t
b r o g r a m w o r k s
t o d a y l s 2 0 1 6 y e a h
```

b) Dla czcionek szeryfowych:

Wykorzystałem czcionkę Times New Roman, w rozmiarze 48px i stworzyłem dla każdej z liter obrazek JPG.



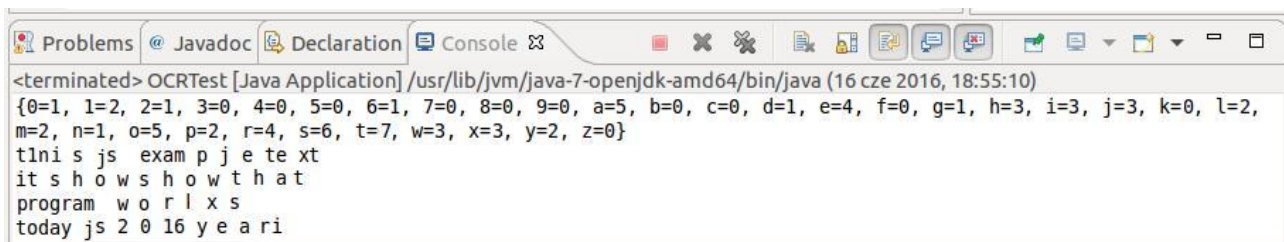
Obraz po którym przeszukiwałem:

W wyniku otrzymałem:

this is example text
it shows how that
program works
today is 2016 year

```
tlni s js exam p j e te xt  
it s h o w s h o w t h a t  
program w o r l x s  
today js 2 0 16 y e a r i
```

I poniżej dołączam cały output jaki otrzymałem w eclipse, łącznie z ilością wystąpień każdego znaku:



```
<terminated> OCRTest [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (16 cze 2016, 18:55:10)  
{0=1, 1=2, 2=1, 3=0, 4=0, 5=0, 6=1, 7=0, 8=0, 9=0, a=5, b=0, c=0, d=1, e=4, f=0, g=1, h=3, i=3, j=3, k=0, l=2,  
m=2, n=1, o=5, p=2, r=4, s=6, t=7, w=3, x=3, y=2, z=0}  
tlni s js exam p j e te xt  
it s h o w s h o w t h a t  
program w o r l x s  
today js 2 0 16 y e a r i
```

3. Wnioski

Niestety choć mój program nie rozpoznaje wszystkich wzorców 100% prawidłowo, to potrafi zidentyfikować poszczególne litery czy słowa. Algorytm niezbyt dobrze rozróżnia gdzie jest spacja, a gdzie przerwa pomiędzy literami. Prawdopodobnie dlatego, że bierze największy z wzorów (u mnie chyba będzie to literka „l” lub „i”) i sprawdza czy odstęp pomiędzy literami jest większy. Jeśli tak, to wstawia spację. W przeciwnym przypadku przechodzi do rozpoznawania kolejnego znaku. Ponadto czasem myli się jeśli pojedyncze litery są blisko siebie lub słowo jest długie.

Ogólnie można powiedzieć, że program dość dobrze sobie radzi z rozpoznawaniem tekstu.

W programie wykorzystano kod źródłowy: <https://github.com/axet/lookup>