



## Zadanie 3 - progowanie

Otwarto: niedziela, 17 marca 2024, 14:15

Progowanie jest najprostszym wariantem segmentacji.

Polega na sprawdzeniu dla każdego punktu obrazu warunku  $L > t$ , czyli czy jasność jest większa od wartości progu  $t$ . Jeśli tak, punkt stanie się biały, jeśli nie, czarny

```
obraz_seg[row,col] = 1 if obraz[row,col] > t else 0
```

Szybciej można zapisać tak

```
obraz_seg = (obraz > t) * 1
```

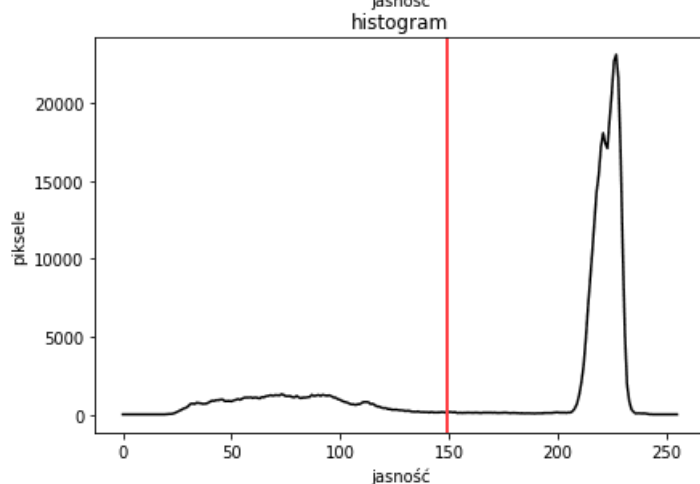
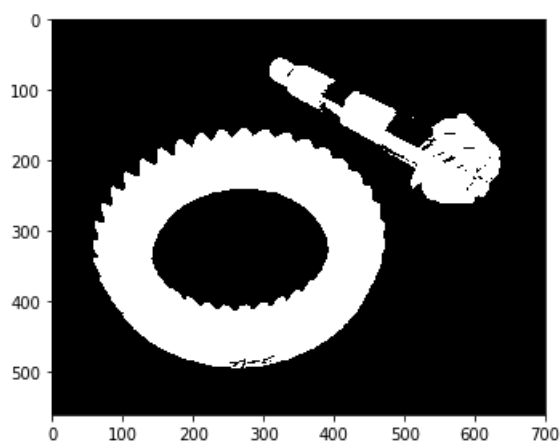
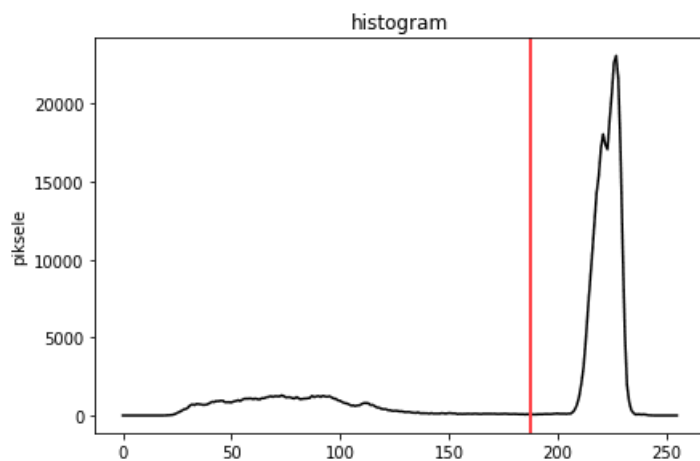
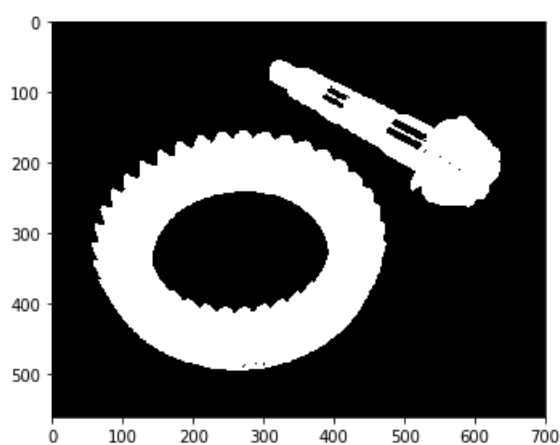
W jego wyniku na obrazie oddzielamy obiekty od tła. Najczęściej obiekty zostaną kolorem białym (1) a tło - czarnym (0). Prawidłowo dobrana wartość  $t$  pozwala oznaczyć obiekty dokładnie tam, gdzie one są.

### ZADANIE 1 (1pkt)

Zapoznaj się z przykładem

[https://scikit-image.org/docs/dev/auto\\_examples/segmentation/plot\\_thresholding.html](https://scikit-image.org/docs/dev/auto_examples/segmentation/plot_thresholding.html)

- Wczytaj obraz *gears1.png*, przekształć jego format do odcieni szarości [0:256].
- Korzystając z gotowej funkcji *try\_all\_threshold* znajdź najlepszą metodę automatyczną. Zanotuj dwie najlepsze metody
- Dwie najlepsze metody zastosuj osobno
- Obraz po zastosowaniu progowania dla obu metod wyświetl wraz z histogramem. Na histogramie zaznacz wartość progu.



## ZADANIE 2

Wczytaj do pamięci obraz *printed\_page.png*.

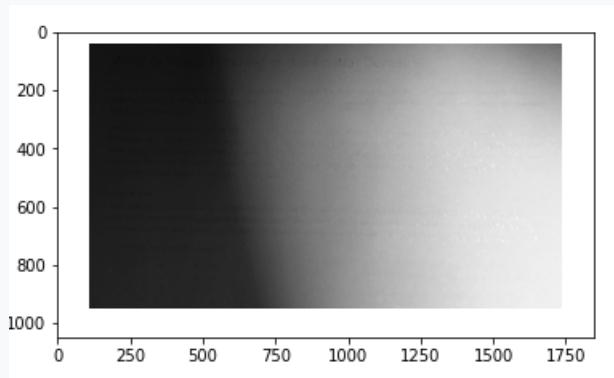
Celem jest wydzielenie tekstu od tła, aby przeprowadzić proces OCR.

Metodą `try_all_threshold` spróbuj którejs z gotowych metod progowania, spróbuj również dobrać ręcznie wartość progu. Efekt może nie być zadowalający.

Obejrzyj przykład zastosowania filtra maximum.

<http://scikit-image.org/docs/dev/api/skimage.filters.rank.html#skimage.filters.rank.maximum>

Korzystając z tego filtra (dobierając odpowiednio wielkość dysku) spróbuj uzyskać obraz, który nazwiesz *tlo*.



Od tego obrazu odejmij obraz oryginalny

```
obraz2 = tlo - obraz
```

Dla otrzymanego obrazu ponownie wykonaj `try_all_threshold` i określ najlepszą metodę.

Wybraną metodę zastosuj (pojedynczo) i zapisz uzyskany obraz na dysku (będzie widać więcej szczegółów niż na figurze. Jeśli efekt nie jest idealny, przed wykonaniem odejmowania przefiltruj tło filtrem `rank.mean`. Właściwy efekt powinien być następujący

## What Is Image Filtering in the Spatial Domain?

Filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement.

Filtering is a *neighborhood operation*, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel. (See *Neighborhood or Block Processing: An Overview* for a general discussion of neighborhood operations.) *Linear filtering* is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood.

### Convolution

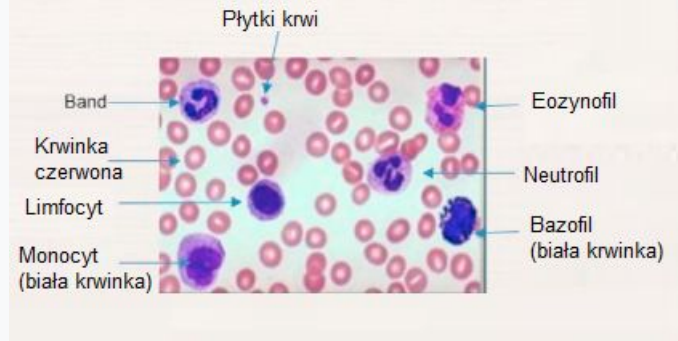
Linear filtering of an image is accomplished through an operation called *convolution*. Convolution is a neighborhood operation in which each output pixel is the weighted sum of neighboring input pixels. The matrix of weights is called the *convolution kernel*, also known as the *filter*. A convolution kernel is a correlation kernel that has been rotated 180 degrees.

For example, suppose the image is

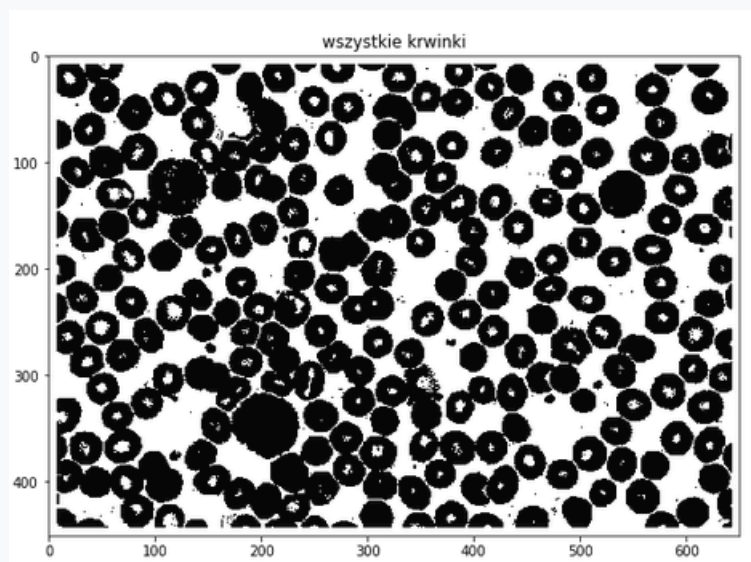
```
A = [17  24   1   8  15
      23   5   7  14  16
        4   6  13  20  22
      10  12  19  21   3
      11  10  17  18  19]
```

## ZADANIE 3

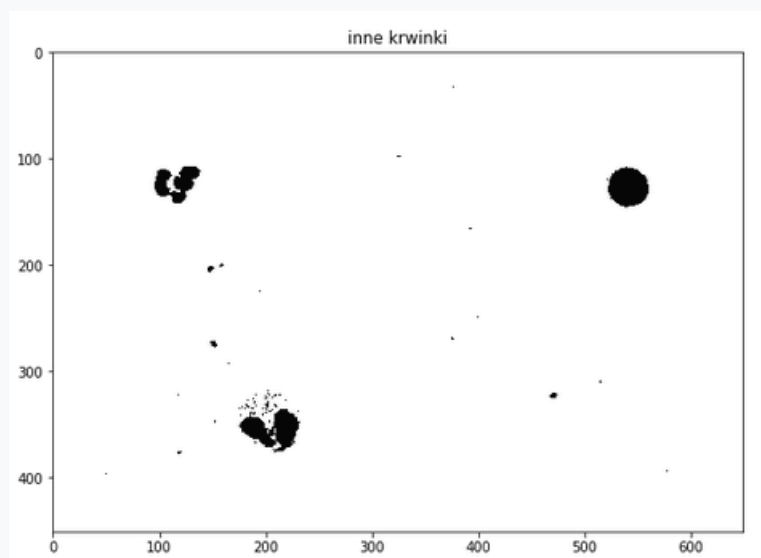
Na obrazie mikroskopowym rozmazu krwi ludzkiej możemy zaobserwować różne komórki



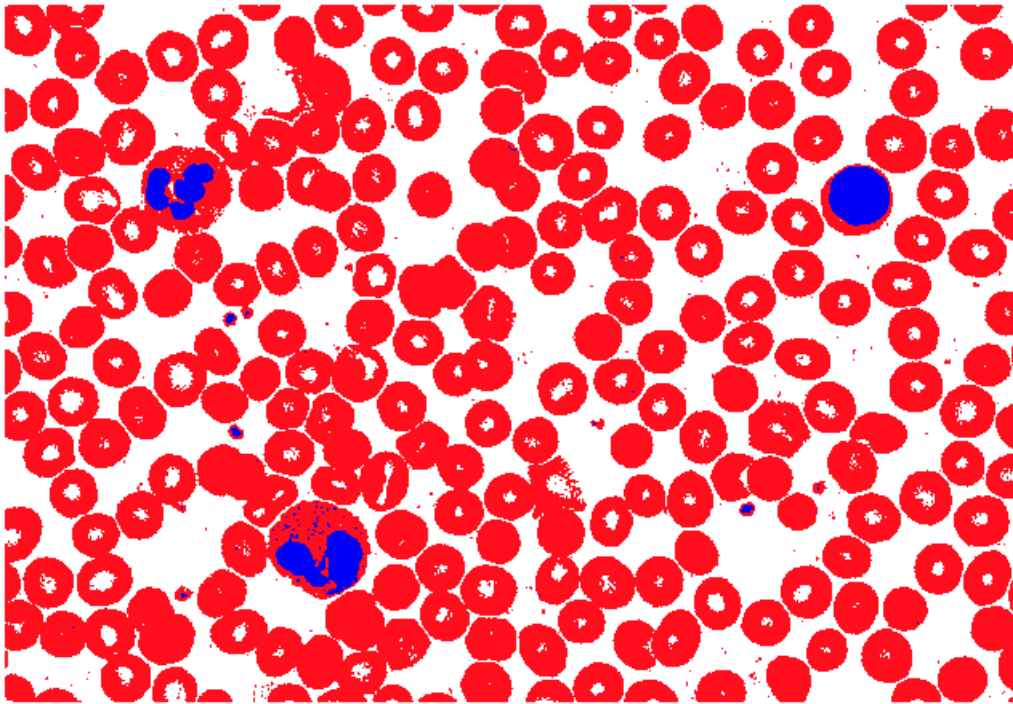
Wczytaj obraz *blood\_smear.jpg* i przekształć go do odcieni szarości. Następnie poprzez ręczny dobór wartości progu *t1* uzyskaj *obraz1* zawierający wszystkie krwinki



Poprzez ręczny wybór wartości progu *t2* uzyskaj *obraz2* zawierający inne krwinki



Mając dobrane wartości *t1* i *t2* utwórz obraz barwny, jak poniżej

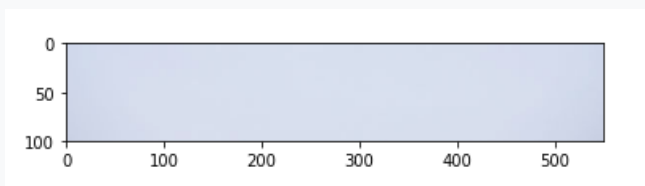


#### ZADANIE 4

a) Korzystając z metody progowania spróbuj uzyskać obraz samolotu *airbus.png*. Zaprezentuj najlepszy wynik, który udało się uzyskać. Mimo wszystko nie będzie on zbyt dobry.

b) Wykonaj progowanie przestrzeni barw następująco:

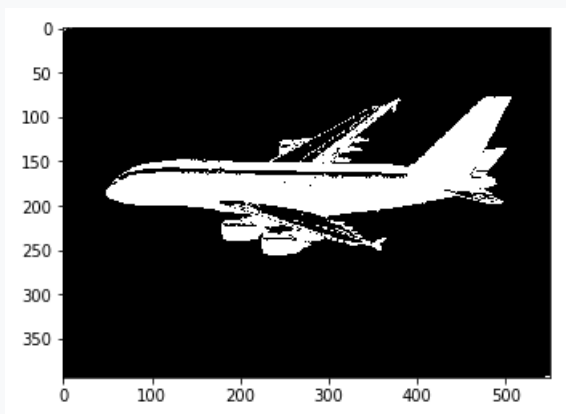
- Wczytaj obraz samolotu, nie przekształcaj go na odcienie szarości.
- Z obrazu wydziel podobraz zawierający tyle dolnych linii żeby nie zawierał on kształtu samolotu i nazwij go np. *niebo*.






- Dla obrazu *niebo* oblicz średnie wartości R,G,B. Skorzystaj z np.average

W pętli po wszystkich wierszach i kolumnach obrazu *niebo* porównaj uzyskaną wartość średnią z wartością dla każdego piksela, traktując kolor jako wektor o współrzędnych (R, G, B). Wykorzystaj np.linalg.norm do obliczenia odległości pomiędzy wektorami. Zapamiętaj maksymalną różnicę jako liczbę *maxDist*.

- W pętli po wszystkich wierszach i kolumnach obrazu *airbus* porównaj każdy piksel obrazu z wartością średnią ponownie korzystając z normy. Jeśli będzie ona większa niż *maxDist* to oznacz piksel jako obiekt (1), w przeciwnym wypadku jako tło (0). Można uzyskać następujący efekt



 [airbus.png](#)  
 [blood\\_smear.jpg](#)  
 [gears1.png](#)  
 [printed\\_text.png](#)

11 marca 2024, 21:28  
 11 marca 2024, 21:28  
 11 marca 2024, 21:28  
 11 marca 2024, 21:28