

---

# Przetwarzanie obrazów cyfrowych :: projekt zaliczeniowy

## SPRAWOZDANIE

---

Autor: Krzysztof Kaczmarek

20.01.2019

### 1. CEL I ZAKRES PROJEKTU

Celem projektu jest zaprojektowanie i zaimplementowanie sekwencji operacji przetwarzania obrazów, która umożliwi obliczenie obiektów (ziaren) znajdujących się na obrazie wejściowym. Zdjęcia zostały zrobione smartfonem Sony Ericsson E2105. Założeniem jest to, że obrazy wejściowe są kolorowe.

Jako ziarna zostały wykorzystane orzeszki ziemne, a ich zdjęcia zostały wykonane na czarnym tle.

Zakres projektu dotyczy:

- a) Załadowania obrazu oryginalnego;
- b) Konwersję obrazu do formatu wieloodcieniowego;
- c) Poprawienie parametrów obrazu;
- d) Binaryzacji;
- e) Wykonania określonych operacji na obrazie binarnym, tak aby pozbyć się z niego drobnych zakłóceń i móc rozdzielić sklejone ze sobą ziarna;
- f) Segmentacji oraz indeksacji obiektów;
- g) Dokonanie pomiarów na obiektach takich jak policzenie ilości obiektów, policzenie jaki procent obszaru zdjęcia zajmują badane obiekty, oraz wyznaczenie ich środków ciężkości.

### 2. OPROGRAMOWANIE

W celu realizacji projektu wykorzystano oprogramowanie Anaconda ver 4.5.12 w raz z Python'em 3.6.6.

Użyto bibliotek:

- a) cv2 – biblioteka funkcji wykorzystywanych podczas obróbki obrazu;
- b) Matplotlib – biblioteka do tworzenia wykresów;
- c) Numpy – biblioteka umożliwiająca zaawansowane obliczenia matematyczne;
- d) Scikit-image – biblioteka zawierająca zbiór algorytmów do przetwarzania obrazu.

Biblioteki te były niezbędne do wykonania czynności opisanych w zakresie projektu. Funkcje zostaną przedstawione poniżej w opisie jego poszczególnych etapów.

### 3. DANE WEJŚCIOWE

Poniżej przedstawione są zdjęcia, które podlegały analizie:



Jak widać na powyższych załącznikach, zdjęcia wykonane były na czarnym tle, ułożenie jak i ilość obiektów była przypadkowa. W niektórych obrazach ziarna są wyraźnie od siebie oddalone, na niektórych są połączone krawędziami. Do zdjęć wykorzystano orzeszki ziemne, porozdzielane na pół, ułożone wybrzuszeniem do góry. Z analizy zdjęć wynika również, że zaistniał problem z odpowiednim naświetleniem zdjęcia (jednak przy szukaniu odpowiedniego miejsca do zrobienia zdjęć okazało się, że to najlepsze możliwe warunki), jak i dodatkowym problemem był pył, który pozostał z orzeszków i soli. Mimo tych niedogodności łatwo odróżnić badane obiekty od tła gołym okiem.

#### 4. OPIS ALGORYTMU

Opis przeprowadzony dla jednego z przykładowych zdjęć

- a) Pierwszym krokiem było zaimportowanie odpowiednich bibliotek oraz obrazu wejściowego do analizy

```
import sys
import cv2
import matplotlib.pyplot as plt
import numpy as np

im = plt.imread('projekt/zdjecia/4.jpg')

plt.figure(figsize=(10,10))
plt.imshow(im, cmap="gray")
plt.axis('off')
plt.show()
```



- b) Wybrany obraz koniecznym było przekonwertować na obraz w skali szarości, by ograniczyć ilość kanałów zdjęcia

```
gray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)

plt.figure(figsize=(10,10))
plt.imshow(gray, cmap="gray")
plt.axis('off')
plt.show()
```



- c) Dla lepszej obserwacji zmian dokonanych przez kolejne operacje zaimplementowana została funkcja pozwalająca na wyświetlenie dwóch obrazów obok siebie

```
def show2imgs(im1, im2, title1='Obraz pierwszy', title2='Obraz drugi', size=(20,20)):

    import matplotlib.pyplot as plt

    f, (ax1, ax2) = plt.subplots(1,2, figsize=size)
    ax1.imshow(im1, cmap='gray')
    ax1.axis('off')
    ax1.set_title(title1)

    ax2.imshow(im2, cmap='gray')
    ax2.axis('off')
    ax2.set_title(title2)
    plt.show()
```

- d) Następnym krokiem było okazanie histogramu zdjęcia w skali szarości by móc ilości punktów na zdjęciu w danej skali szarości

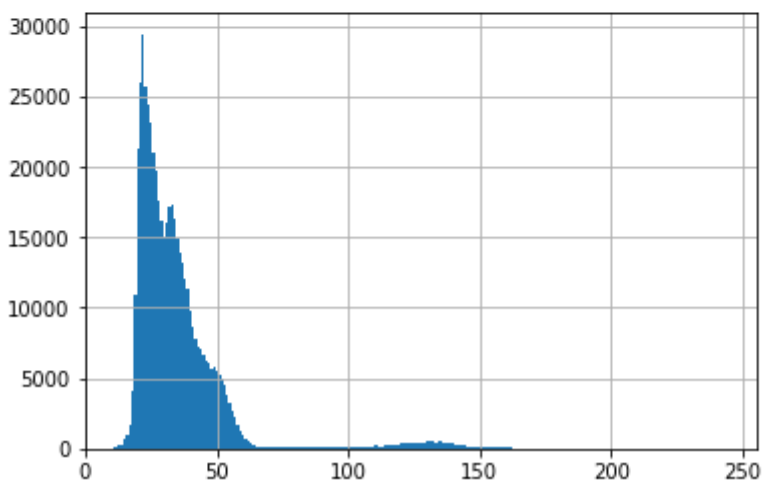
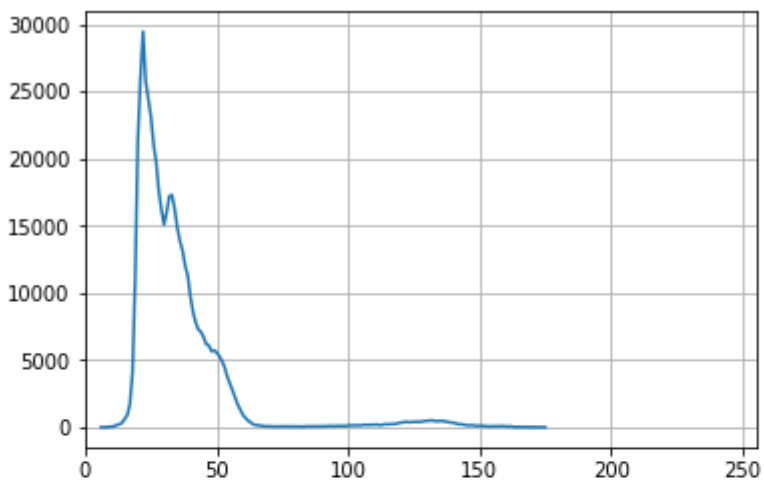
```
from skimage import exposure

histogram = exposure.histogram(gray, nbins=256)
hist, cbins = histogram

plt.plot(cbins, hist)
plt.xlim([-0.2, 255])
plt.grid()
plt.show()

plt.bar(cbins, hist, width=1)
plt.xlim([-0.2, 255])
plt.grid()
plt.show()
```

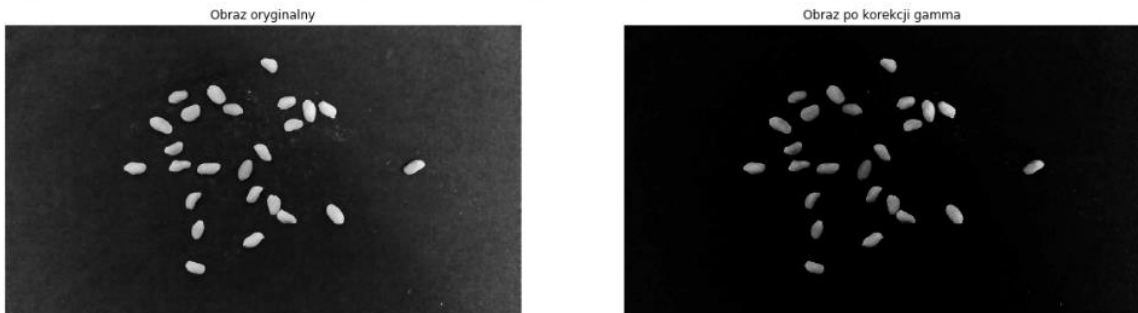
Histogramy (wykresy liniowy i słupkowy):



- e) Jak widać na zdjęciu (jak i histogramach) czarne tło okazało się nie być całkowicie czarne, tylko ciemnoszare. Najciemniejszy obszar jest na środku, po bokach widać nieco jaśniejsze tło (skutek średniej jakości oświetlenia jak i cień rzucany przy robieniu zdjęć). Dlatego też konieczne było poprawienie parametrów zdjęcia, by jak najbardziej przybliżyć tło do czarnego.

Zastosowano korekcję gamma zdjęcia.

```
gamma = 2.5
gimf = exposure.adjust_gamma(gray, gamma=gamma)
show2imgs(gray, gimf, title1='Obraz oryginalny', title2='Obraz po korekcji gamma', size=(20,20))
```



Widać od razu, że obraz wynikowy jest zdecydowanie ciemniejszy od oryginalnego, ale jednocześnie można dalej łatwo zauważyć obiekty na zdjęciu.

- f) Następnym krokiem było zastosowanie binaryzacji, by móc otrzymać obraz dwutonowy, na którym łatwiej można dokonywać dalszej analizy. Zastosowano tutaj binaryzację OTSU z progiem automatycznym.

```
ret,thresh = cv2.threshold(gimf,0,255,cv2.THRESH_OTSU)
show2imgs(gimf, thresh, title1='Obraz po korekcji gamma', title2='Obraz po binaryzacji', size=(20,20))
```



Otrzymano obraz „czarno-biały”, lecz jeszcze wymagający dalszego przetwarzania ze względu na drobiny występujące wśród obiektów.

- g) Następnym krokiem było użycie operacji morfologicznych na zdjęciu, takich jak operacja otwarcia (dla pozbycia się szumów i drobin, oraz dla oddzielenia krawędzi od połączonych obiektów), znalezienie obszaru tła obiektów za pomocą operacji dyatacji, znalezienie obszaru zajętego przez obiekty za pomocą transformacji odległościowej, oraz wyznaczenie obszaru nieznanego, który nie jest przyporządkowany ani do obiektów, ani do ich tła.

Kod źródłowy:

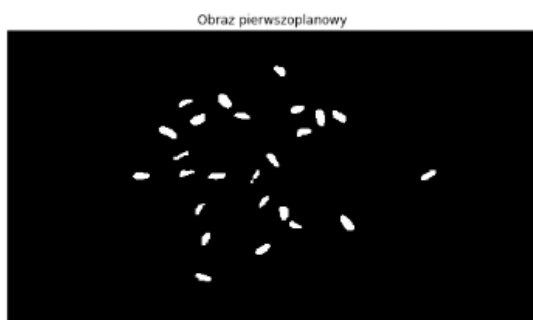
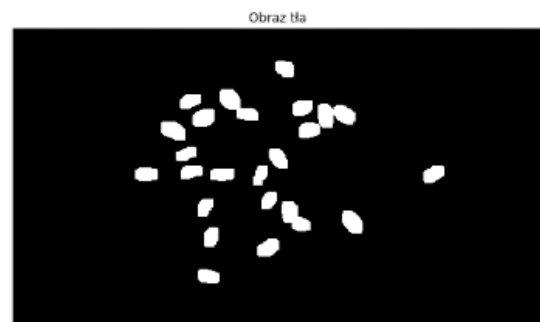
```
# otwarcie
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 4)

# pewny obszar tła
sure_bg = cv2.dilate(opening,kernel,iterations=3)

# Pewny obszar pierwszoplanowy (obektów)
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform,0.3*dist_transform.max(),255,0)

# Obszar nieznan
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_fg)
```

Oraz obrazy wynikowe:





Dzięki tym operacjom będzie można zastosować operację watershed, czyli będziemy mogli dokonać segmentacji wododziałowej na obiektach.

- h) Następny krok dotyczy zdefiniowania znaczników pozwalających na oznaczenie i segmentację oraz indeksację obiektów.

```
# Etykietowanie znaczników
ret, markers = cv2.connectedComponents(sure_fg)

# Dodanie 1 do etykiet, dzięki czemu nasze tło będzie miało wartość 1, a nie 0
markers = markers+1

# Oznaczamy obszar nieznany jako 0
markers[unknown==255] = 0

markers = cv2.watershed(im, markers)
im[markers == -1] = [255,0,0]
```

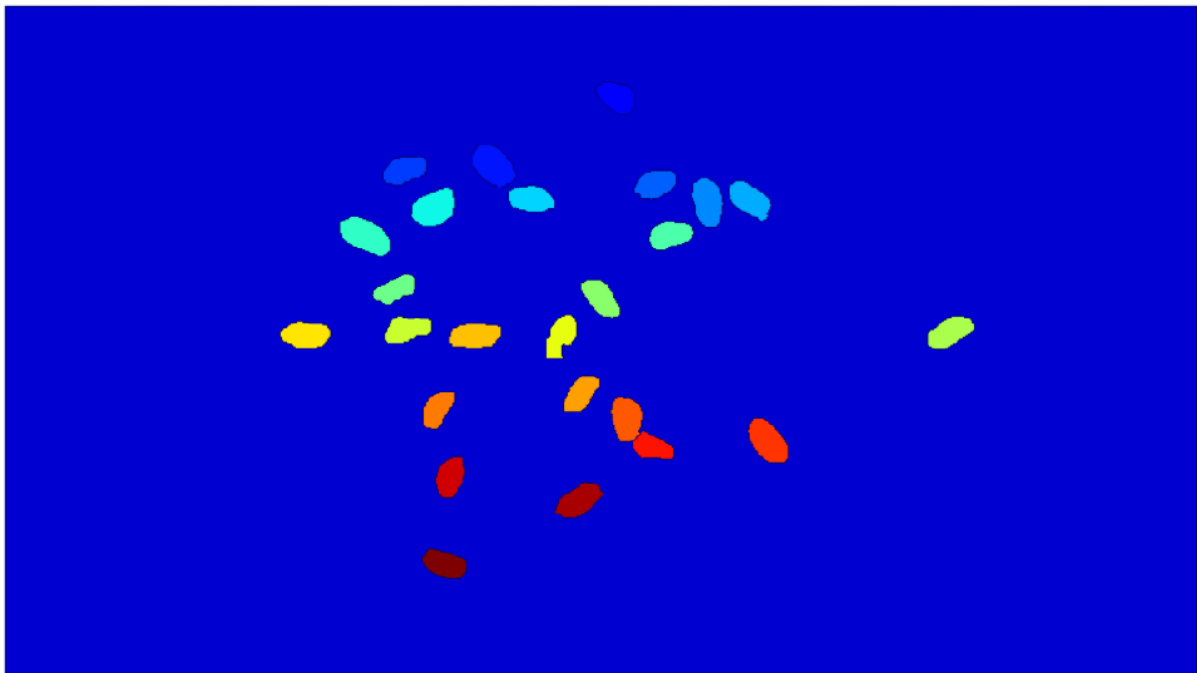
Osobno oznaczone zostały obiekty, tło jak i obszar nieznany, następnie wykonano wspomnianą wyżej operację watershed.



Jak widać, na obrazie, poszczególne elementy zostało oznaczone osobnym znacznikiem (obrysem), w tym również tło obrazu.



To samo zdjęcie pokazane w barwie kolorów:



Jak widać, segmentacja i indeksacja przeprowadzona została prawidłowo, ponieważ każdy z obiektów jest oznaczony inną barwą. Nawet obiekty połączone ze sobą krawędzie udało się od siebie rozdzielić.

- i) Po zakończonej indeksacji można było już w prosty sposób wyliczyć ukazane obiekty na obrazie.

```
number_seeds = len(np.unique(markers))-2  
print ('number of seeds : %i' % number_seeds)
```

```
number of seeds : 25
```

Ilość obiektów na zdjęciu oraz wyliczona przez algorytm jest poprawna.

j) Kolejnym krokiem było obliczenie jaki procent całego zdjęcia stanowią obiekty

```
#sumowanie punktów zajętych przez obiekty
suma=0
for i in range(nb_labels):
    pts = getFigure(label_objects, i+1)
    suma+=len(pts)

from skimage import feature, measure

#ilościowy i procentowy rozmiar zdjęcia
width, height = opening.shape[:2]
img_count = width*height
img_percent = img_count*100.0/width/height
print('Ilość punktów na zdjęciu wynosi:', img_count)
print('Procentowy udział: ',img_percent)

properties = measure.regionprops(label_objects)
label_areas = [int(prop.area) for prop in properties]

#ilościowy i procentowy rozmiar zdjęcia po odjęciu sumy punktów zajętych przez obiekty
imga_count = width*height-suma
imga_percent = imga_count*100.0/width/height
print('Ilość punktów na zdjęciu po odjęciu sumy punktów zajmowanej przez obiekty:', imga_count)
print('Procentowy udział: ',imga_percent)

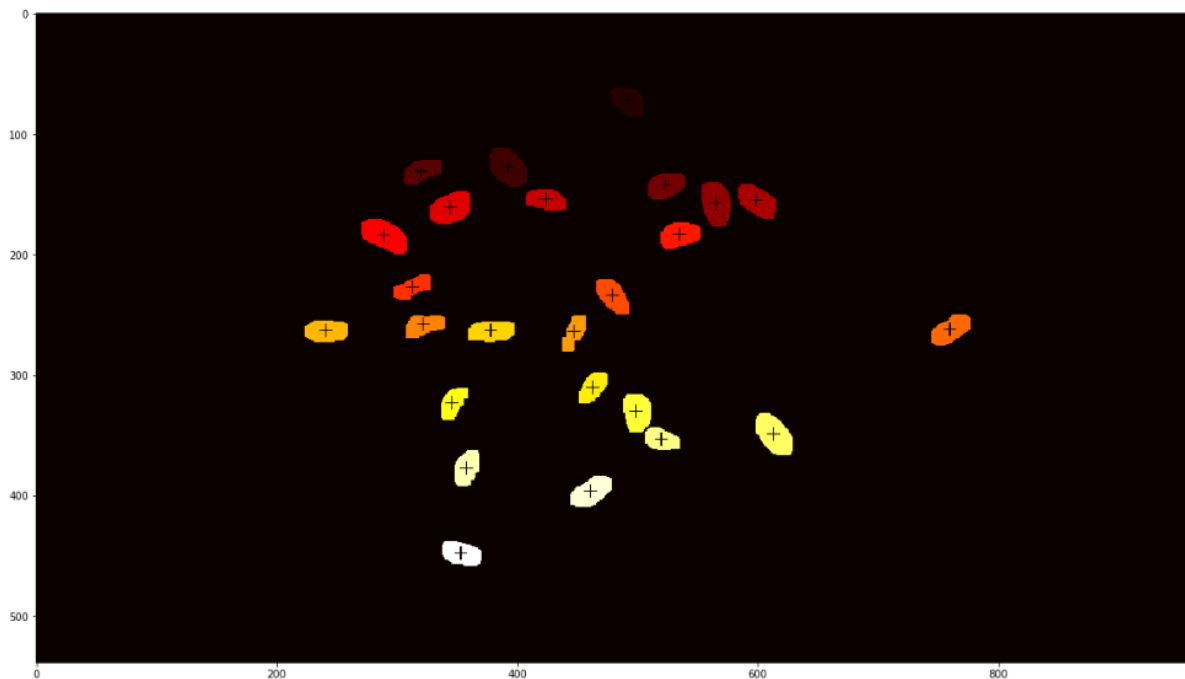
#Obliczenie udziału procentowego
procentx = img_percent - imga_percent
print ('Procentowy udział obiektów na zdjęciu:',procentx)
```

Wyniki obliczeń:

```
Ilość punktów na zdjęciu wynosi: 517440
Procentowy udział: 100.0
Ilość punktów na zdjęciu po odjęciu sumy punktów zajmowanej przez obiekty: 503260
Procentowy udział: 97.2595856524428
Procentowy udział obiektów na zdjęciu: 2.7404143475572056
```

Udział obiektów na zdjęciu wynosi 2,74%, co na pierwszy rzut oka nie jest widoczne. Jednak jeśli obiekty zostałyby rozłożone na pojedyncze piksele i ułożone na obrazie wzdłuż jednej z krawędzi zdjęcia, byłoby to bardziej zauważalne.

h) Następnie wyznaczone zostały środki ciężkości obiektów i naniesione na obraz



Poniżej wypisane zostały liczby punktów zajętych przez poszczególne obiekty jak i ich środki ciężkości (pierwsza współrzędna dotyczy osi y, druga osi x)




Liczba punktów:	526	Srodek ciezkosci:	[72.83269961977186, 492.25665399239546]
Liczba punktów:	761	Srodek ciezkosci:	[127.63600525624179, 392.5755584756899]
Liczba punktów:	479	Srodek ciezkosci:	[131.01878914405012, 320.36743215031316]
Liczba punktów:	555	Srodek ciezkosci:	[142.93693693693695, 523.5351351351351]
Liczba punktów:	732	Srodek ciezkosci:	[157.61065573770492, 565.4931693989071]
Liczba punktów:	649	Srodek ciezkosci:	[155.6979969183359, 599.6471494607088]
Liczba punktów:	486	Srodek ciezkosci:	[154.6172839506173, 424.11728395061726]
Liczba punktów:	710	Srodek ciezkosci:	[161.4943661971831, 344.75070422535214]
Liczba punktów:	776	Srodek ciezkosci:	[184.36211340206185, 289.8492268041237]
Liczba punktów:	579	Srodek ciezkosci:	[183.83074265975822, 535.2158894645942]
Liczba punktów:	415	Srodek ciezkosci:	[227.0843373493976, 313.255421686747]
Liczba punktów:	561	Srodek ciezkosci:	[234.53297682709447, 479.6167557932264]
Liczba punktów:	579	Srodek ciezkosci:	[262.41623488773746, 760.8756476683938]
Liczba punktów:	480	Srodek ciezkosci:	[258.95, 322.675]
Liczba punktów:	389	Srodek ciezkosci:	[264.81748071979433, 447.5038560411311]
Liczba punktów:	572	Srodek ciezkosci:	[263.52797202797206, 241.54195804195805]
Liczba punktów:	573	Srodek ciezkosci:	[263.7766143106457, 378.5828970331588]
Liczba punktów:	434	Srodek ciezkosci:	[310.10599078341016, 463.3041474654378]
Liczba punktów:	428	Srodek ciezkosci:	[323.2172897196262, 346.8481308411215]
Liczba punktów:	630	Srodek ciezkosci:	[330.8857142857143, 499.86349206349206]
Liczba punktów:	768	Srodek ciezkosci:	[349.51171875, 613.7955729166666]
Liczba punktów:	434	Srodek ciezkosci:	[353.6774193548387, 520.1082949308756]
Liczba punktów:	503	Srodek ciezkosci:	[377.46918489065604, 358.013916500994]
Liczba punktów:	620	Srodek ciezkosci:	[396.81451612903226, 461.72741935483873]
Liczba punktów:	541	Srodek ciezkosci:	[448.1293900184843, 353.68761552680223]




i) Na sam koniec obliczono wariancję dla środków ciężkości (osobno dla x i y)



Wariancja srodków ciężkości: [47.699892014273914, 17.633582590903565]

## 5. WYNIKI EKSPERYMENTALNE


W poniższej tabeli przedstawiono wyniki analizy poszczególnych zdjęć przepuszczonych przez algorytm.

	Ilość obiektów na zdjęciu: 11	Uwagi: Wynik prawidłowy
	Wyliczona ilość obiektów przez algorytm: 11	
	Poprawność algorytmu: 100%	
	Procent zdjęcia zajęty przez obiekty: 1,56%	
	Ilość obiektów na zdjęciu: 25	Uwagi: Wynik prawidłowy
	Wyliczona ilość obiektów przez algorytm: 25	
	Poprawność algorytmu: 100%	
	Procent zdjęcia zajęty przez obiekty: 2,61%	
	Ilość obiektów na zdjęciu: 19	Uwagi: Analiza problemu zapisana we wnioskach (pkt. 6)
	Wyliczona ilość obiektów przez algorytm: 20	
	Poprawność algorytmu: 94,74%	
	Procent zdjęcia zajęty przez obiekty: 2,51%	

	Ilość obiektów na zdjęciu: 49	Uwagi: Wynik prawidłowy
	Wyliczona ilość obiektów przez algorytm: 49	
	Poprawność algorytmu: 100%	
	Procent zdjęcia zajęty przez obiekty: 5,98%	
	Ilość obiektów na zdjęciu: 25	Uwagi: Wynik prawidłowy
	Wyliczona ilość obiektów przez algorytm: 25	
	Poprawność algorytmu: 100%	
	Procent zdjęcia zajęty przez obiekty: 2,74%	
	Ilość obiektów na zdjęciu: 51	Uwagi: Wynik prawidłowy
	Wyliczona ilość obiektów przez algorytm: 51	
	Poprawność algorytmu: 100%	
	Procent zdjęcia zajęty przez obiekty: 7,14%	

	Ilość obiektów na zdjęciu: 58	Uwagi: Wynik prawidłowy
	Wyliczona ilość obiektów przez algorytm: 58	
	Poprawność algorytmu: 100%	
	Procent zdjęcia zajęty przez obiekty: 6,97%	
	Ilość obiektów na zdjęciu: 25	Uwagi: Wynik prawidłowy
	Wyliczona ilość obiektów przez algorytm: 25	
	Poprawność algorytmu: 100%	
	Procent zdjęcia zajęty przez obiekty: 3,23%	
	Ilość obiektów na zdjęciu: 20	Uwagi: Wynik prawidłowy
	Wyliczona ilość obiektów przez algorytm: 20	
	Poprawność algorytmu: 100%	
	Procent zdjęcia zajęty przez obiekty: 2,41%	



	Ilość obiektów na zdjęciu: 25	Uwagi: Wynik prawidłowy
	Wyliczona ilość obiektów przez algorytm: 25	
	Poprawność algorytmu: 100%	
	Procent zdjęcia zajęty przez obiekty: 2,24%	

Całkowita szacunkowa poprawność działania programu na badanych zdjęciach wynosi: **99,47%**.

## 6. WNIOSKI

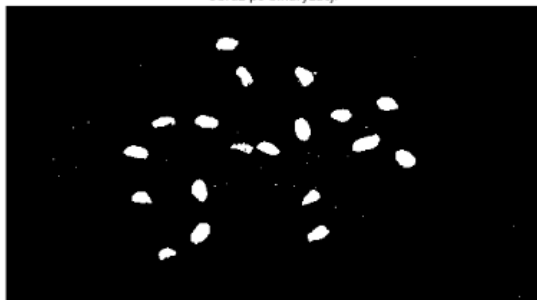
W większości przypadków program zadziałał bez większych zarzutów. Program był w stanie praktycznie bez problemu oddzielić obiekty od tła, policzyć je, jak i wyznaczyć ich środki ciężkości. Natomiast w jednym z przypadków nie zadziałał on do końca prawidłowo, i to na zdjęciu, gdzie obiekty nie są ze sobą połączone. Na zdjęciu przedstawiony jest problematyczny obiekt (zauważalnie – ciemniejszy od pozostałych).





Sekwencja obrazów (wyznaczanie obiektów):

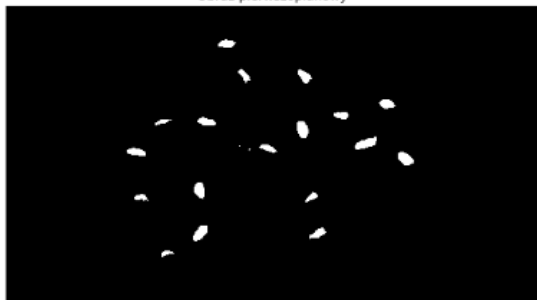
Obraz po binaryzacji



Obraz po otwarciu



Obraz pierwszoplanowy



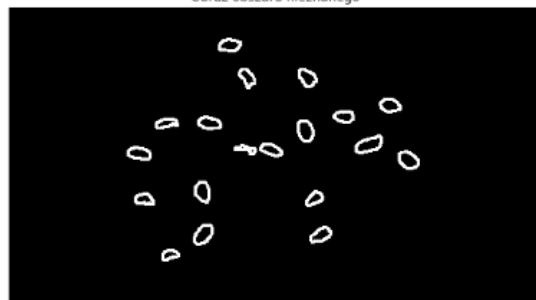
Obraz tła



Obraz pierwszoplanowy



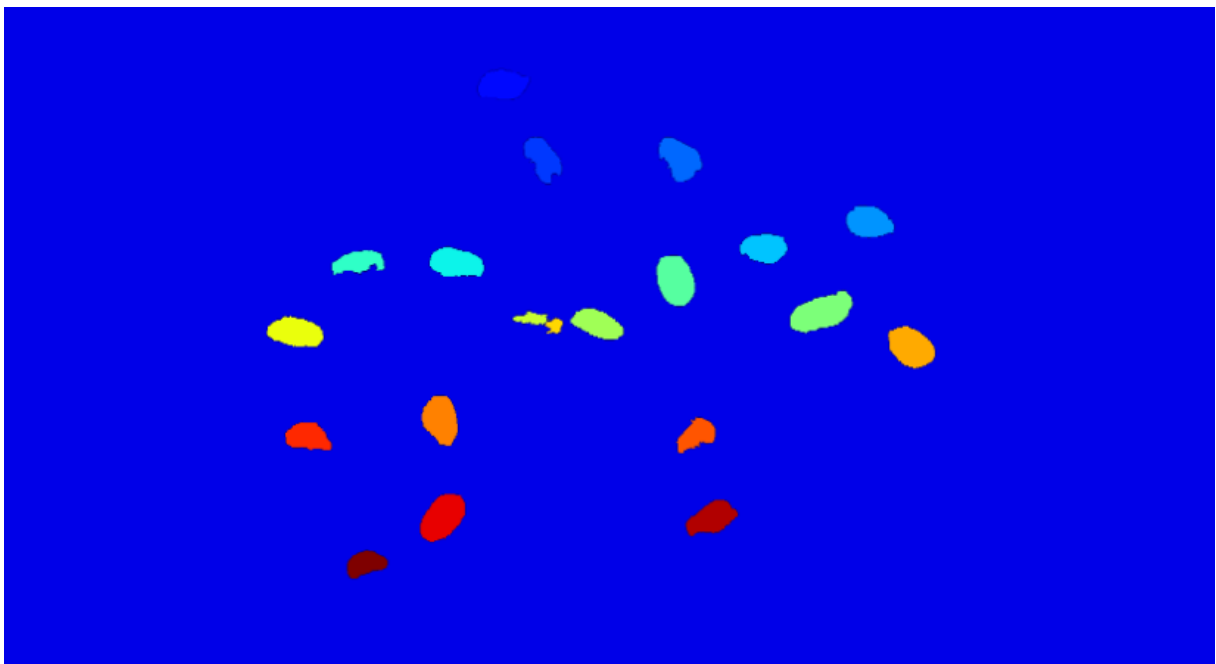
Obraz obszaru nieznanego



Jak widać na powyższym zdjęciu, obiekt praktycznie zniknął z obrazu, przez co w dalszej analizie wygląda tak:



Na powyższym zdjęciu że w miejscu jednego obiektu zostały wykryte dwa obiekty. Dokładniej przedstawione jest na obrazie poniżej.



W celu pozbycia się problemu można przetestować różne parametry dotyczące korekcji gamma, binaryzacji i otwarcia. Zmienianie parametrów po testach poprawiło algorytm, niestety pogarszając operacje oddzielania krawędzi dla pozostałych zdjęć, przez co została podjęta decyzja o nie zmienianiu parametrów.

Niestety, jakość zdjęć i oświetlenia pozostawia wiele do życzenia. Zdjęcia robione starym smartfonem są bardzo niskiej jakości, chociaż mimo tego problemu wszelkie szumy jak i pyłki zostały usunięte ze

zdjęć. Jednym z mankamentów robienia zdjęć smartfonem jest to że mimo robienia ich w tej samej pozycji i wysokości, to obiekty na nich mają czasem różny rozmiar. Prawdopodobne jest, że jeśli zdjęcia zostały by wykonane lepszej jakości aparatem to problem by nie zaistniał.

Na szczęście nie było problemów z wyliczeniem obiektów o tej samej wielkości, nawet jeśli stykały się one krawędziami.

Jednak przy niskiej jakości zdjęć i zakłóceniach na nich występujących poprawność wykonania programu wynosi 99,47%, co stanowi bardzo dobry wynik. Algorytm został przetestowany na dziesięciu różniących się zdjęciach, dlatego program można używać do badania innych obiektów ze zdjęć przy założeniu w miarę kontrastowego tła, ewentualnie zmieniając korekcję gamma w zależności od oświetlenia zdjęcia, czy też zmieniając współczynnik przy transformacji odległościowej.