RAPORT Z PROJEKTU

Przetwarzanie Informacji Wizyjnej

Szybkie zliczanie/etykietowanie obiektów w obrazie binarnym

Grupa dziekańska **TI-3** Rok akademicki **2021/2022.** Semestr **VII**

Data wykonania raportu **18.10.2021.** Nr sekcji**: 44** Nr tematu **26**.

Skład sekcji:

Albert Pintera

Jakub Kaniowski

Cel projektu

Celem projektu jest zaproponowanie oraz implementacja algorytmu zliczającego lub kolorującego białe obiekty na obrazie binarnym. W angielskiej literaturze określa się to zagadnienie terminem: *binary image connected components algorithm* lub *labeling algorithm*. Implementacja może zostać zrealizowana w dowolnym środowisku programistycznym.

Wykonany program

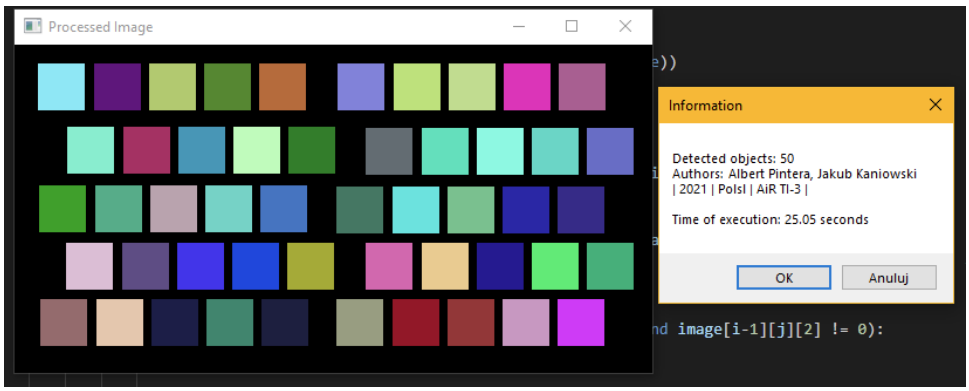
Do stworzenia programu realizującego zadanie projektowe zostało wykorzystane środowisko Python wraz z biblioteką OpenCV. Plikiem wejściowym jest obraz binarny przedstawiający białe obiekty na czarnym tle. Zaimplementowany algorytm opiera się na analizie i przetworzeniu kolejnych pixeli.

Działanie zastosowanego algorytmu można przedstawić w kilku następujących po sobie krokach:

1. Wczytanie obrazu
2. Sprawdzenie wymiarów obrazu
3. Kolorowanie białych obszarów wiersz po wierszu (generowanie nowych kolorów)
4. Zmiana koloru danego pixela na podstawie analizy jego otoczenia.
5. Ponowna analiza otoczenia danego pixela (od prawej do lewej obrazu) w celu połączenia sąsiadujących ze sobą kolorów w konkretnym obiekcie.
6. Zliczenie obiektów w oparciu o liczbę wszystkich występujących kolorów w obrazie.
7. Prezentacja wyników na ekranie.

Prezentacja działania

Interfejs programu prezentuje się następująco:



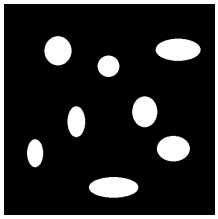
Okno “Processed Image” wyświetla pokolorowany obraz (detekcja obiektów białych). Okno “Information” zawiera wynik działania programu – wykryte obiekty oraz czas wykonywania się programu.

Dodatkowo, zdublowana informacja wyświetla się w konsoli systemowej.

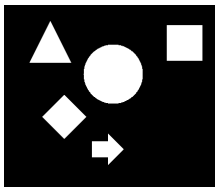
```
PS M:\PIW> & C:/Users/Kuba/AppData/Local/Programs/Python/Python38/python.exe m:/PIW/main.py


Program info:

Detected objects: 50
Authors: Albert Pintera, Jakub Kaniowski
| 2021 | Polsl | AiR TI-3 |

Time of execution: 25.02 seconds
```

Wygenerowano kilka testowych obrazów:

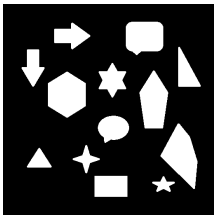
Shapes.png



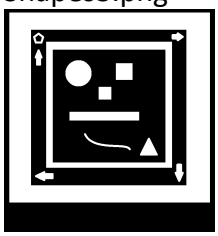
Shapes1.png



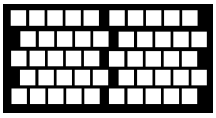
Shapes2.png



Shapes3.png

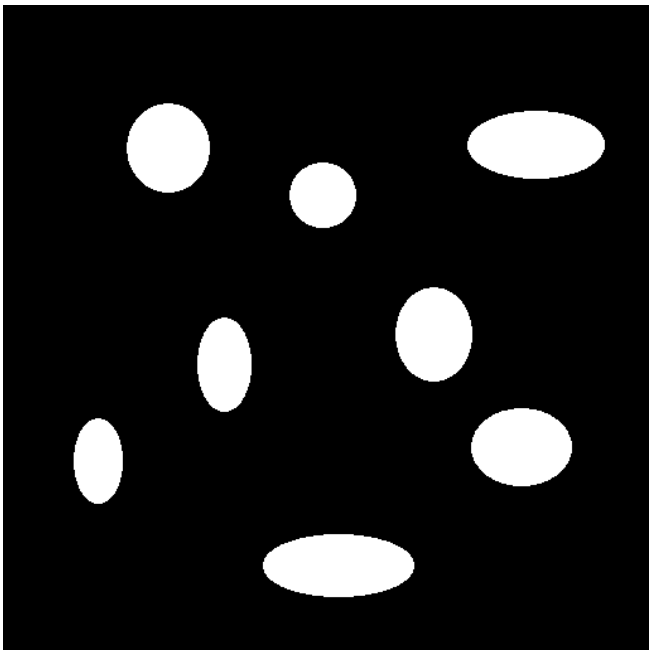


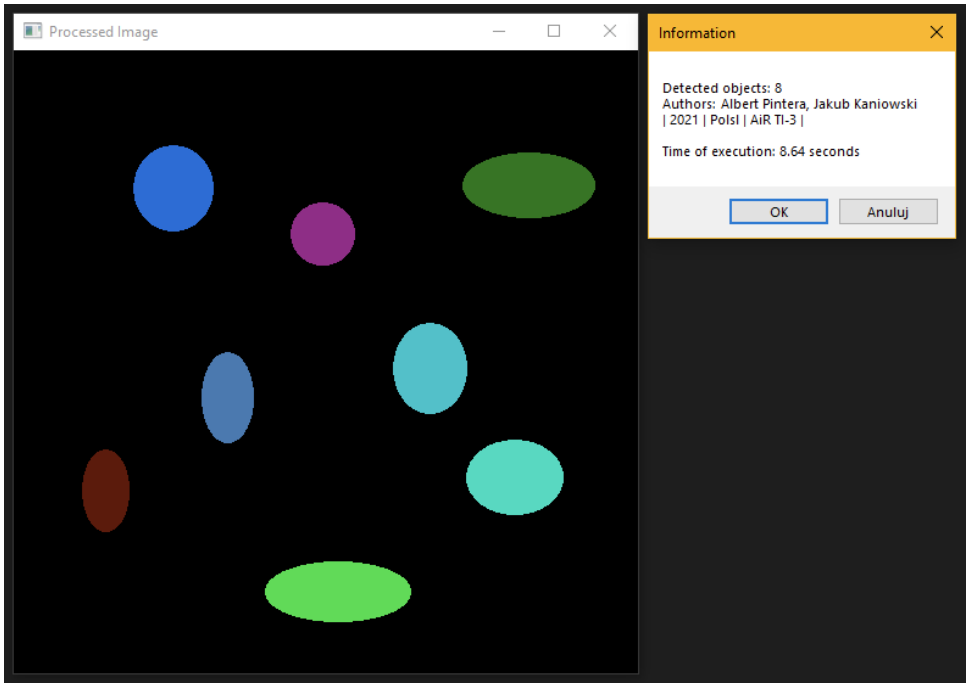
Shapes4.png



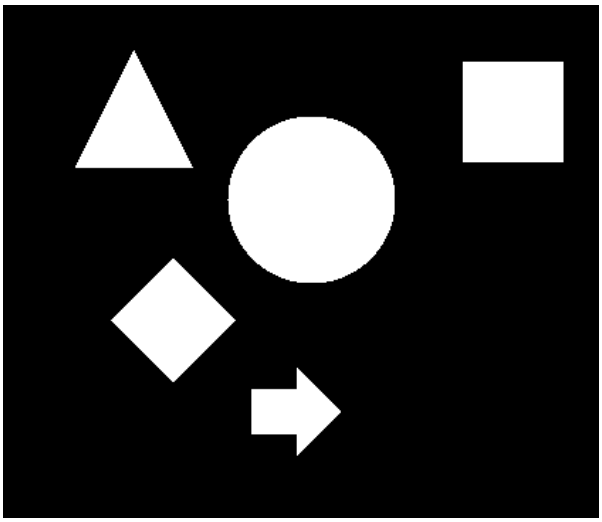
Wyniki działania programu:

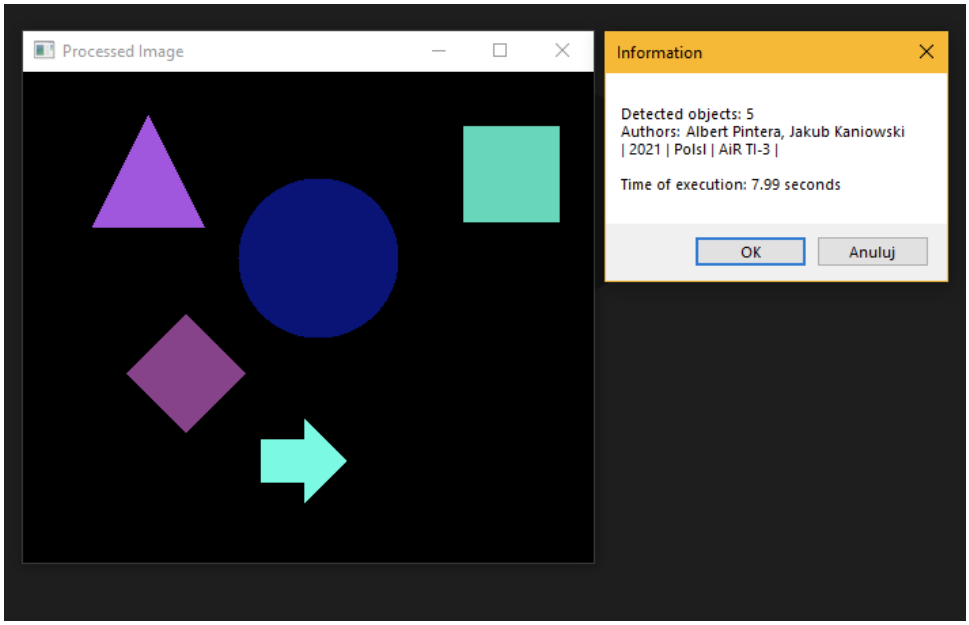
Obraz Shapes.png. Obrazów rzeczywistych: 8. Wynik działania programu: 8 obiektów



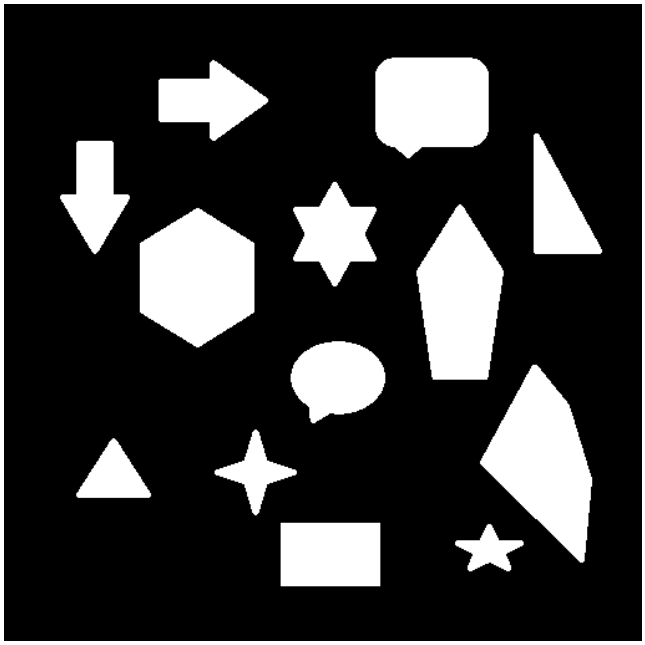


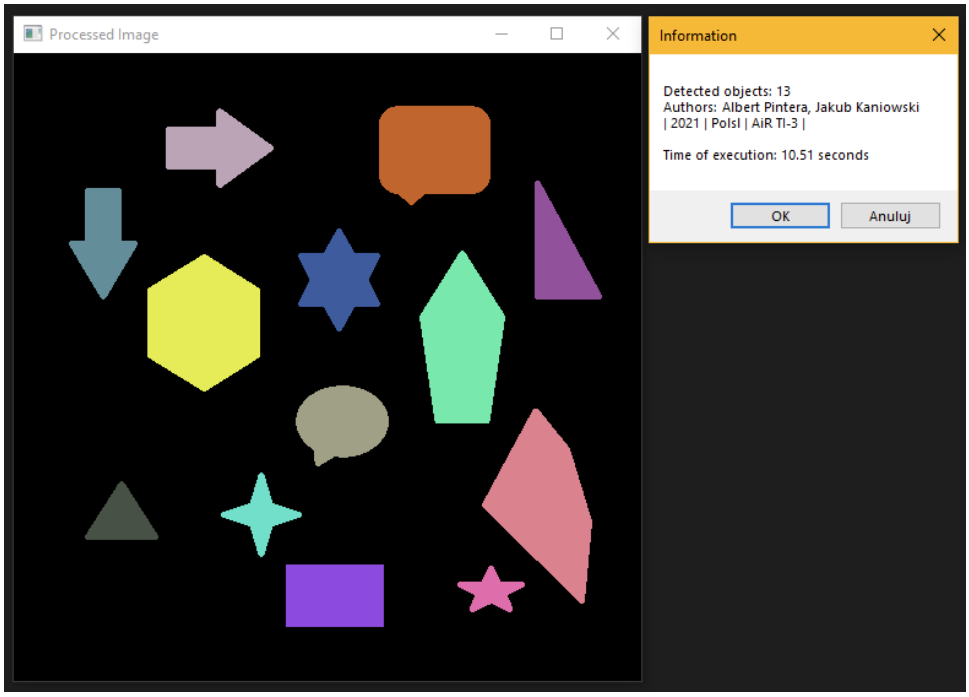
Obraz Shapes1.png. Obrazów rzeczywistych: 5. Wynik działania programu: 5 obiektów



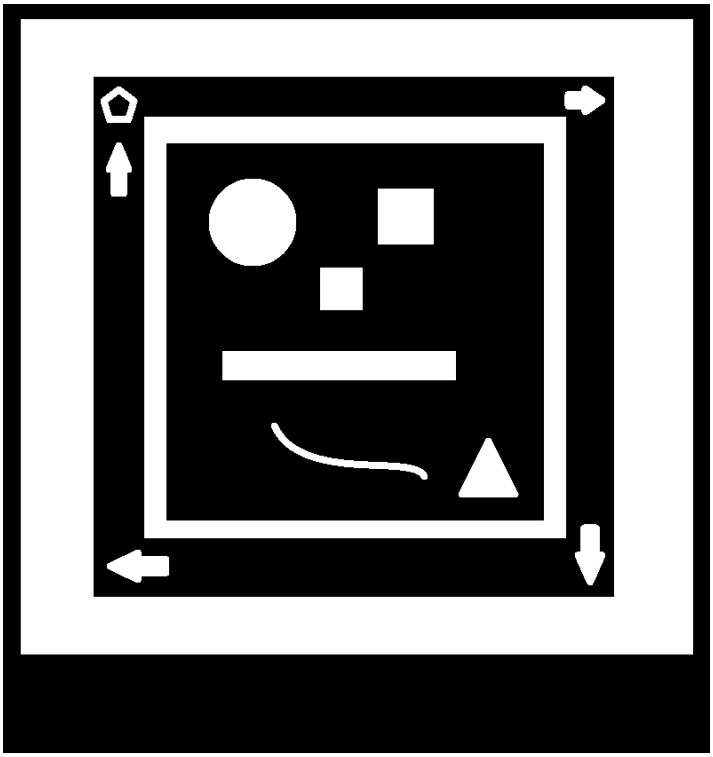


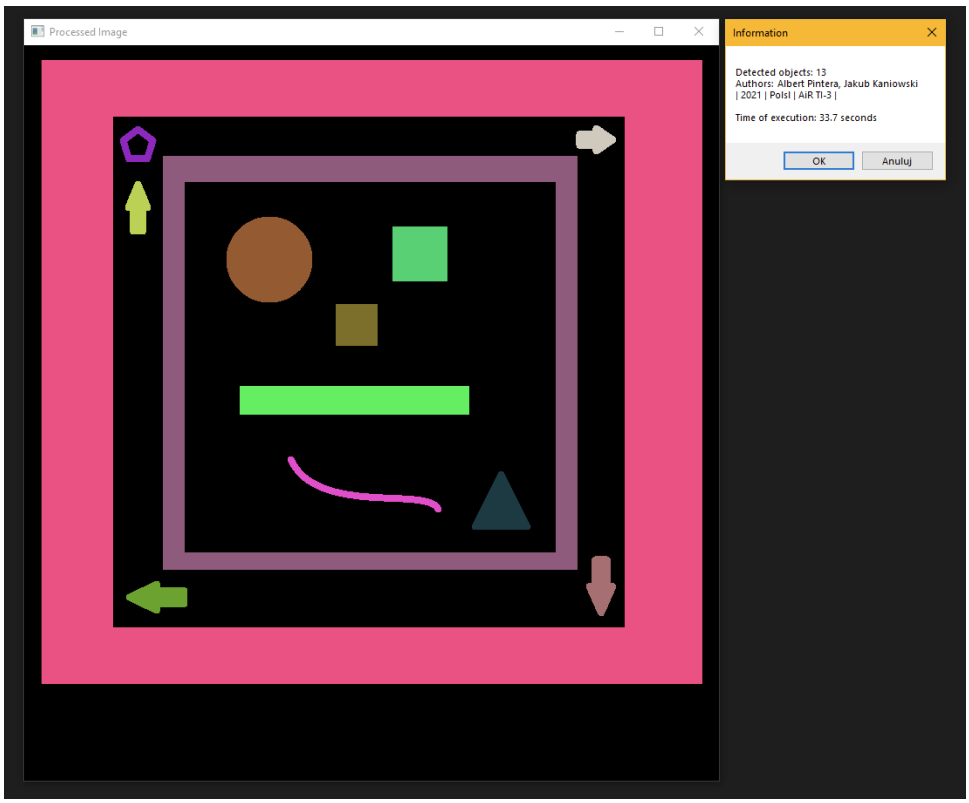
Obraz Shapes2.png. Obrazów rzeczywistych: 13. Wynik działania programu: 13 obiektów



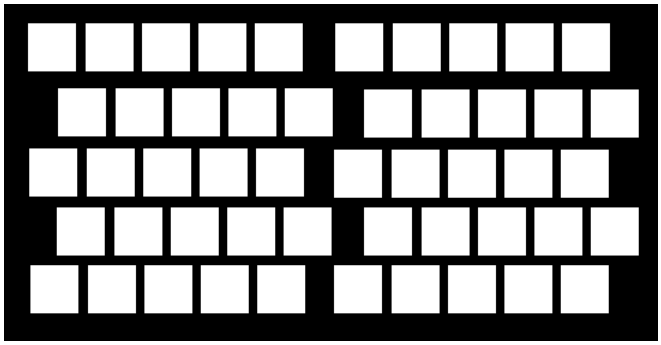


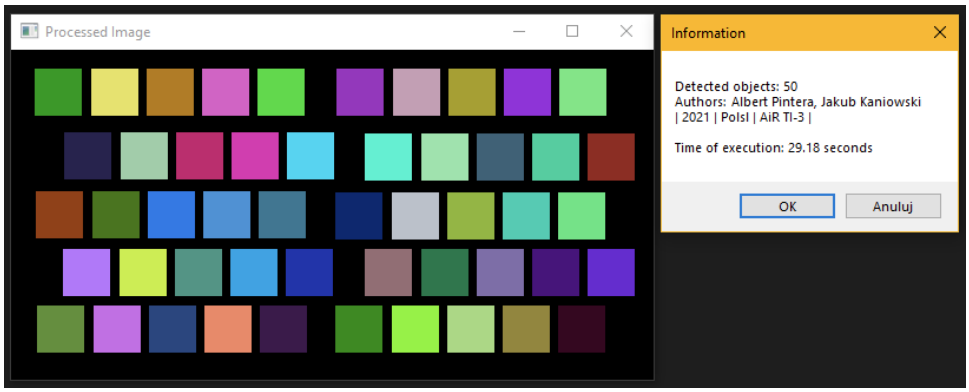
Obraz Shapes3.png. Obrazów rzeczywistych: 13. Wynik działania programu: 13 obiektów





Obraz Shapes4.png. Obrazów rzeczywistych: 50. Wynik działania programu: 50 obiektów





Wnioski

Program wykonany w ramach projektu spełnia stawiane mu wymagania. Algorytm koloruje obiekty na obrazie, ale również zwraca ich liczbę. Zaimplementowane operacje wykonują się stosunkowo szybko, jednakże czas ten zależy w dużej mierze od rozmiaru oraz złożoności obrazu wejściowego (tj. Liczby białych i czarnych elementów). Działanie programu zweryfikowano wykorzystując metodę biblioteki OpenCV *cv2.connectedComponents(),* która stanowi funkcję realizującą algorytm etykietowania. Wyniki gotowej funkcji porównywane z napisanym programem były spójne, co oznacza poprawne działanie.

Kod źródłowy

Wersja czytelna dostępna w załącznikach (main.py)

```
# Politechnika Slaska - Wydzial Automatyki Elektroniki i Informatyki

# Autorzy: Albert Pintera, Jakub Kaniowski

# 2021



import cv2

import random

import numpy as np

import ctypes

import time


start_time = time.time()     #Rozpoczecie zliczania czasu wykonania programu


counter_of_objects = 0       #Licznik obiektow

list_of_colors = []          #Lista kolorow


#Lista kolorow - pierwszy nowy kolor

new_color = list(np.random.choice(range(1, 255, 1), size=3, replace=False))


image = cv2.imread('shapes.png', 0)                 #Wczytanie obrazu (sciezka)

image = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)      #Zamiana barw na obrazie
```

```
#Wymiary obrazu

rows = image.shape[0]

cols = image.shape[1]



cols1 = image.shape[1]-1



#Funkcja losujaca kolejne kolory

def generateColor(rng, i):

    global counter_of_objects

    return (np.random.choice(range(1,(rng-1),1), size=i, replace=False))



for i in range(rows):

    for j in range(cols):



        if image[i][j][0] == 255 and image[i][j][1] == 255 and image[i][j][2] == 255:
# 255 - bialy kolor

            image[i][j] = new_color



            if image[i][j+1][0] == 0 and image[i][j+1][1] == 0 and image[i][j+1][2] == 0:
# 0 - czarny kolor

                new_color = generateColor(256,3)
#Losuje nowy kolor jak znajdzie nowy bialy kolor



            if(i>0):

                if (image[i-1][j][0] != 0 and image[i-1][j][1] != 0 and image[i-1][j][2] !=
0):          #Algorytm rozglada sie wyzej, na poprzedni wiersz od aktualnego miejsca

                    image[i][j] = image[i-1][j]



                elif(image[i][j-1][0] != 0 and image[i][j-1][1] != 0 and image[i][j-1][2] !=
0):         #Algorytm rozglada sie w lewo od aktualnego miejsca

                    image[i][j] = image[i][j-1]
```

```
                elif(image[i-1][j-1][0] != 0 and image[i-1][j-1][1] != 0 and image[i-1][j-1][2] != 0):  #Algorytm rozglada sie w lewo w poprzednim wierszu od aktualnego miejsca (lewy gorny rog)

                    image[i][j] = image[i-1][j-1]



                elif(image[i-1][j+1][0] != 0 and image[i-1][j+1][1] != 0 and image[i-1][j+1][2] != 0):  #Algorytm rozglada sie w prawo w poprzednim wierszu od aktualnego miejsca (prawy gorny rog)

                    image[i][j] = image[i-1][j+1]



    #Ponowny odczyt wiersza, z przeciwnej strony (od prawej do lewej - ujednolica kolory)

    for k in range(cols1):

        if(k<cols1):

            if(image[i][cols1-k-1][0] == 0 and image[i][cols1-k-1][1] == 0 and image[i][cols1-k-1][2] == 0):

                continue

            if((image[i][cols1-k][0] != 0 and image[i][cols1-k][1] != 0 and image[i][cols1-k][2] != 0) and (image[i][cols1-k][0] != 255 and image[i][cols1-k][1] != 255 and image[i][cols1-k][2] != 255)):  #patrz na prawo jeśli kolor nie jest czarny lub bialy to wtedy trzeba sie cofnac kolorkami

                image[i][(cols1-k)-1] = image[i][cols1-k]




#Ponowny odczyt calego obrazu w celu zliczenia wystepujacych obiektow

for i in range(rows):

 for j in range(cols):

  if (image[i][j][0] != 255 and image[i][j][1] != 255 and image[i][j][2] != 255) and (image[i][j][0] != 0 and image[i][j][1] != 0 and image[i][j][2] != 0):

            if any(np.array_equal([image[i][j][0],image[i][j][1],image[i][j][2]], x) for x in list_of_colors):

                continue

            else:


list_of_colors.append(np.array([image[i][j][0],image[i][j][1],image[i][j][2]]))



counter_of_objects = len(list_of_colors)    #Przypisanie liczby zliczonych kolorow do zmiennej przechowujacej liczbe obiektow
```

```
#Wyswietlenie uzyskanych wynikow

cv2.imshow('Processed Image', image)



#Dla kazdego systemu

print("\n\nProgram info: \n\nDetected objects: " + str(counter_of_objects) + "\nAuthors:
Albert Pintera, Jakub Kaniowski\n| 2021 | Polsl | AiR TI-3 |" + "\n\nTime of execution: " +
str(float("{0:.2f}".format(time.time() - start_time))) + " seconds")



#Dla windowsa (okomentowac w razie potrzeby)

ctypes.windll.user32.MessageBoxW(0, "Detected objects: " + str(counter_of_objects) +
"\nAuthors:\tAlbert Pintera, Jakub Kaniowski\n| 2021 | Polsl | AiR TI-3 |" + "\n\nTime of
execution: " + str(float("{0:.2f}".format(time.time() - start_time))) + " seconds",
"Information " , 1)



cv2.waitKey(0)

cv2.destroyAllWindows()
```