Obraz zawierający tekst

Opis wygenerowany automatycznie

**Automatyczne rozpoznawanie tablic rejestracyjnych w pojazdach**

**Dr inż. Leszek Gajecki Karol Drewniak**

**Sylwester Włodyga**

**Maciej Szlag**

**Rzeszów 2022**

Spis treści

[Początek kodu. 3](#_Toc95407855)

[Określanie regionu tablicy rejestracyjnej 4](#_Toc95407856)

[Wczytywanie pliku zdjęcia. 5](#_Toc95407857)

[Konwertowanie z RGB do kolorów szarości. 6](#_Toc95407858)

[Bilateral Filter 6](#_Toc95407859)

[Usuwanie szumu za pomocą filtru bilateral 6](#_Toc95407860)

[Znajdowanie krawędzi w szarym obrazie 7](#_Toc95407861)

[Znajdowanie konturów 8](#_Toc95407862)

[Kontury na podstawie ich powierzchni. 9](#_Toc95407863)

[Pętla for, aby znaleźć najlepszy możliwy przybliżony kontur tablicy rejestracyjnej 10](#_Toc95407864)

[Używamy tesseractu, aby przekonwertować obraz na ciąg 12](#_Toc95407865)

[Kosztorys 13](#_Toc95407866)

[Wnioski 13](#_Toc95407867)

[Bibliografia: 14](#_Toc95407868)

## Początek kodu.

W pierwszym kroku musimy załadować plik i go wyświetlić.

***#import th necessary packages***

***import cv2***

***import argparse***

***#load imge from disk via "cv2.imread" and then grab the spatial***

***#dimension, inc. width, height, and number of channels***

***img = cv2.imread('WOT32488.jpg',0)***

***image = img = cv2.imread('WOT32488.jpg')***

***(h, w, c) = image.shape[:3]***

***# display the image width, height, and number of channels to our***

***# terminal***

***print("width: {} pixels".format(w))***

***print("height: {} pixels".format(h))***

***print("channels: {}".format(c))***

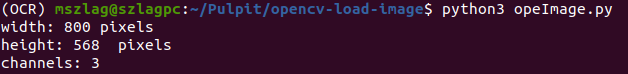
***# show the image and wait for a keypress***

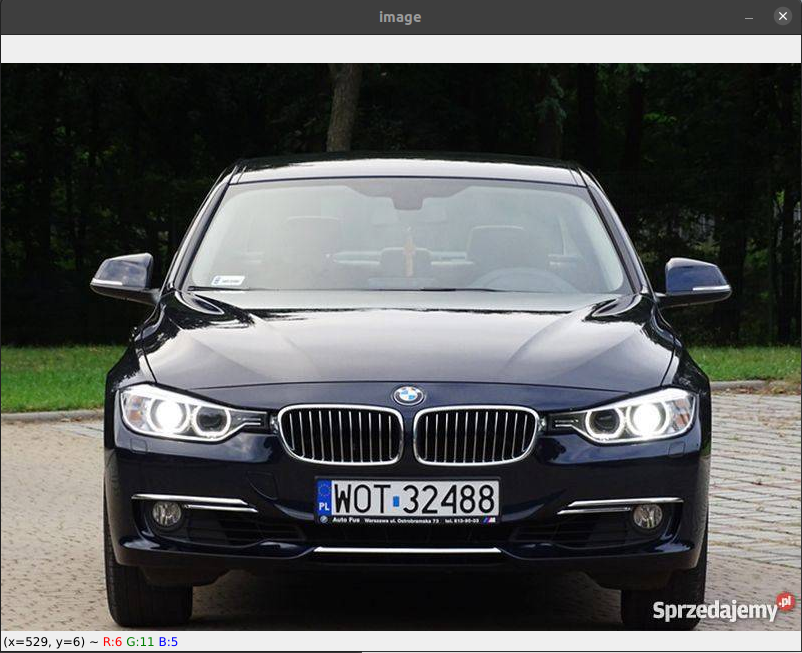
***cv2.imshow('image',img)***

***cv2.waitKey(0)***

Naciśnij F2, aby zapisać, lub wpisz yes i naciśnij enter a następnie dwukrotnie przycisk Esc, aby powrócić do terminala.

W terminalu wpisz polecenie: python3 opeImage.py, aby uruchomić nasz skrypt, dane wyjściowe pokazują dane i wyzwolony obraz



****

## Określanie regionu tablicy rejestracyjnej

Naszym celem jest wykrywanie tablic rejestracyjnych z oryginalnego zdjęcia przy użyciu technik przetwarzania obrazu krawędzi. Korzystając z tych technik, przetwarzamy obraz binarny uzyskany poprzez identyfikację punktów o ostrych zmianach jasności w stosunku do oryginalnego obrazu. Taka zmiana jest bardziej zauważalna i częstsza niż w innych częściach obrazu. Prostokątny wykres rejestracyjny ma bardziej wyraźne cechy krawędzi i tekstury niż inne obszary obrazu. Oprócz zalet metoda ta ma też wady - po prostu nie nadaje się do zdjęć z nadmierną ilością krawędzi.

**Teraz zajmiemy się pierwszym krokiem, czyli wykryciem i lokalizacją tablicy rejestracyjnej ze zdjęcia.**

W pierwszej części obraz jest przetwarzany w celu wyeliminowania zniekształceń, detali i innych niepotrzebnych elementów. Wykorzystano do tego celu bibliotekę OpenCV, która jest bogatym zestawem funkcji do szybkiego przetwarzania obrazu. Biblioteka posiada otwarty kod źródłowy (Open Source).

## Wczytywanie pliku zdjęcia.

***#Import bibliotek:***

|  |
| --- |
| ***import numpy as np*** |

|  |
| --- |
| ***import cv2*** |

|  |
| --- |
| ***import imutils*** |

***import pytesseract***

**#Kolejnym krokiem jest załadowanie pliku:**

img = cv2.imread(‘lpd/plate.jpg’)

**#Teraz musimy zmienić długość:**

img = imutils.resize(img, width=550)

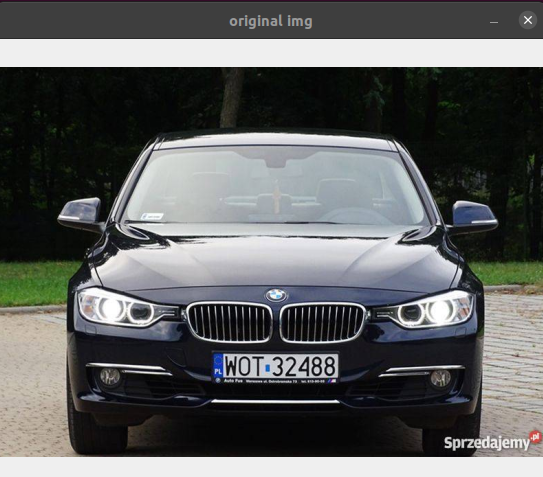
**#Wyświetlamy oryginalny obraz:**

**cv2.imshow(“original img”, img)**

**#I oczekiwanie na wywołanie funkcji zamknięcia**

**cv2.waitKey(0)**

**W rezultacie otrzymujemy:**

****

## Konwertowanie z RGB do kolorów szarości.

|  |
| --- |
| ***# RGB to Gray scale conversion*** |

|  |
| --- |
| ***gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)*** |

|  |
| --- |
| ***cv2.imshow("1 – BRG to Grey scale Conversion", gray)*** |

***cv2.waitKey(0)***

In the ouput:



### 

## Bilateral Filter

Iteracyjne usuwanie szumów filtru dwustronnego: podstawowym celem filtrowania jest usunięcie szumu i zniekształceń z obrazu. Podczas robienia zdjęć przez aparat oraz z powodu warunków pogodowych mogą wystąpić szumy. W proponowanej metodzie redukcji szumów zastosowano iteracyjny filtr dwukierunkowy. Iteracyjny filtr dwustronny jest filtrem nieliniowym. Zapewnia mechanizm redukcji szumów, zachowując jednocześnie krawędzie skuteczniej niż filtr medianowy.

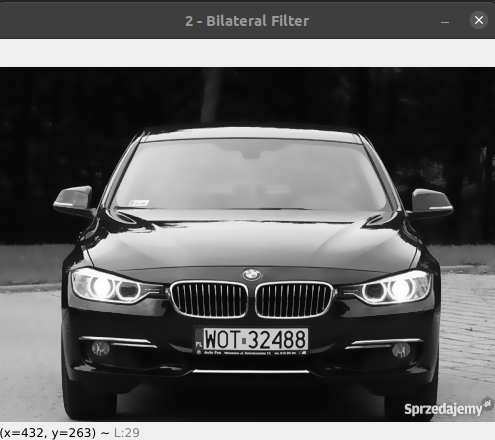
## Usuwanie szumu za pomocą filtru bilateral

***gray = cv2.bilateralFilter(gray, 11, 17, 17)***

***cv2.imshow("2 - Bilateral Filter", gray)***

***cv2.waitKey(0)***

Output file:



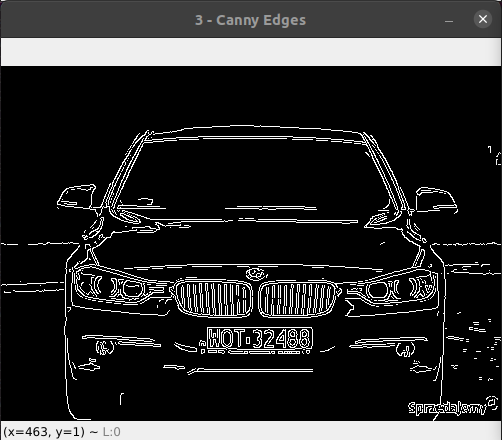
### 

## Znajdowanie krawędzi w szarym obrazie

Wykrywanie krawędzi jest szeroko stosowane w wykrywaniu tablic rejestracyjnych jako technika przetwarzania wstępnego. Są to filtry zwiększające wydajność obrazu, służą do wykrywania krawędzi, co jest ważne przy wykrywaniu tablic rejestracyjnych.

|  |
| --- |
| ***edged = cv2.Canny(gray, 170, 200)*** |

|  |
| --- |
| ***cv2.imshow("3 - Canny Edges", edged)***  ***cv2.waitKey(0)***  Output file: |

****

### 

## Znajdowanie konturów

Znajdowanie wszystkich konturów na obrazie wejściowym. Funkcja cv2.findContours zwraca wszystkie kontury znalezione na obrazie. Kontury można wyjaśnić po prostu jako krzywą łączącą wszystkie ciągłe punkty (wzdłuż granicy) o tym samym kolorze lub intensywności.

# Znajdowanie konturu bazując na krawędzi

*cnts, new = cv2.findContours(edged.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)*

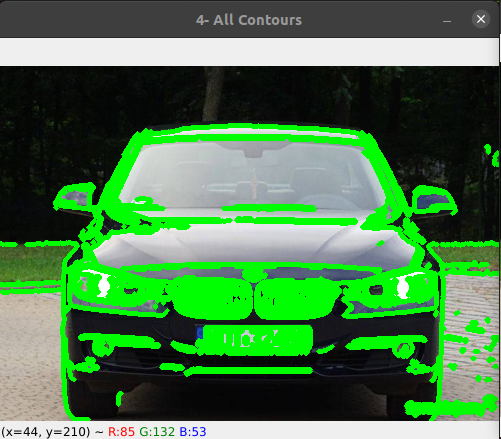
*# Create copy of original image to draw all contours*

*img1 = img.copy()*

*cv2.drawContours(img1, cnts, -1, (0,255,0), 3)*

*cv2.imshow("4- All Contours", img1)*

*cv2.waitKey(0)*

**Output File:**

## Kontury na podstawie ich powierzchni.

Utrzymanie minimalnego wymaganego obszaru jako „30” (wszystko mniejsze niż to nie będzie brane pod uwagę)

***cnts=sorted(cnts, key = cv2.contourArea, reverse = True)[:30]***

***NumberPlateCnt = None #Krak konturu tablicy***

# Top 30 Contours

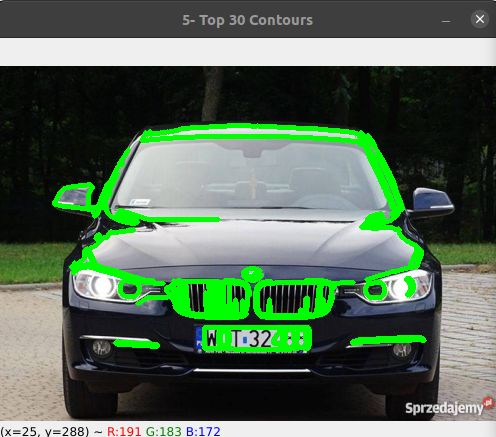
*img2 = img.copy()*

*cv2.drawContours(img2, cnts, -1, (0,255,0), 3)*

*cv2.imshow("5- Top 30 Contours", img2)*

*cv2.waitKey(0)*

output file:

****

## Pętla for, aby znaleźć najlepszy możliwy przybliżony kontur tablicy rejestracyjnej

*count = 0*

*idx =7*

*for c in cnts:*

*peri = cv2.arcLength(c, True)*

*approx = cv2.approxPolyDP(c, 0.02 \* peri, True)*

*# print ("approx = ",approx)*

*if len(approx) == 4: # Select the contour with 4 corners*

*NumberPlateCnt = approx #This is our approx Number Plate Contour*

# Wycięcie zdjęcia z konturem i zapisanie

*x, y, w, h = cv2.boundingRect(c) #This will find out co-ord for plate*

*new\_img = gray[y:y + h, x:x + w] #Create new image*

*cv2.imwrite('Cropped\_img/' + str(idx) + '.png', new\_img) #Store new image*

*idx+=1*

break

# PRzedstawienie obrysu na oryginalnym zdjęciu

print(NumberPlateCnt)

*cv2.drawContours(image, [NumberPlateCnt], -1, (0,255,0), 3)*

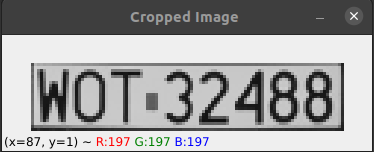
*cv2.imshow("Final Image With Number Plate Detected", image)*

*cv2.waitKey(0)*



*Cropped\_img\_loc = 'Cropped\_img/7.png'cv2.imshow("Cropped Image ", cv2.imread(Cropped\_img\_loc))*

output file:

****

## Używamy tesseractu, aby przekonwertować obraz na ciąg

***text = pytesseract.image\_to\_string(Cropped\_img\_loc, lang='eng' config=’--psm 6’)***

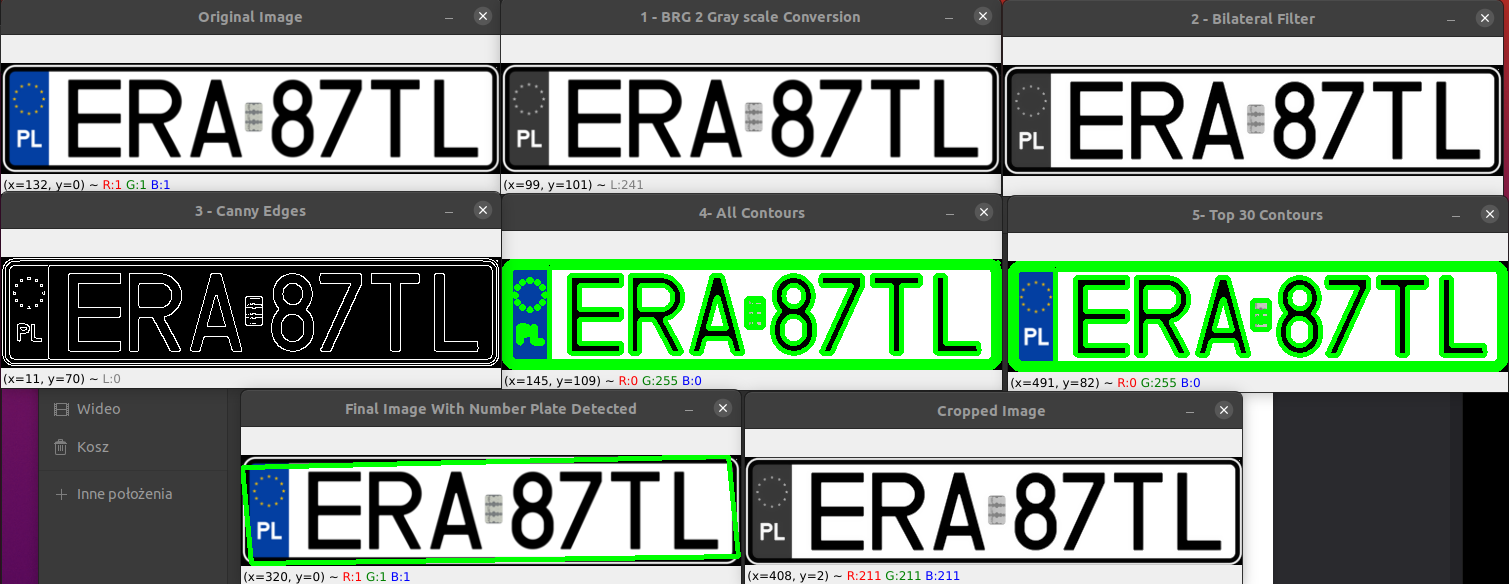
***print("Number is :", text)***

***Output:***

***Number is : WOT32488***

***Przedstawienie procesu krok po kroku:***

Numer to : WOT32488

***Number is: ERA87L***

## Kosztorys

|  |  |  |  |
| --- | --- | --- | --- |
| LP. | Nazwa przedmiotu/czynności | Ilość (szt.) | Kwota (zł - brutto) |
| 1. | [Raspberry Pi 4 model B WiFi DualBand Bluetooth 4GB RAM 1,5GHz](https://botland.com.pl/moduly-i-zestawy-raspberry-pi-4b/16579-raspberry-pi-4-model-b-wifi-dualband-bluetooth-8gb-ram-15ghz-765756931199.html) | 1 | 219,00 |
| 2. | Kamera HD B OV5647 5Mpx - z regulacją ostrości dla Raspberry Pi - Waveshare 8193 | 1 | 77,00 |
| 3. | Karta pamięci justPi microSD 32GB 100MB/s klasa 10 + system NOOBs do Raspberry Pi 4B/3B+/3B/2B | 1 | 29,90 |
| 4. | Zasilacz USB C 5,1V / 3A do Raspberry Pi 4 oryginalny | 1 | 47,90 |
| Suma: | | | 373,80 |

Urządzenie na którym został napisany kod nie jest wliczone w kosztorys ponieważ, w dzisiejszych czasach każdy posiada w domu podstawowy komputer klasy laptop lub klasy PC, który w zupełności wystarczy ze swoimi parametrami do instalacji oprogramowania oraz środowiska do pisania kodu. Kolejnym elementem, który został pominięty jest literatura. W dobie internetu mamy dostępem do masy podstawowych artykułów, szkoleń czy tutoriali.

## Wnioski

ANPR, który zużywa mniej czasu przetwarzania, mniej mocy obliczeniowej i ma lepsze wskaźniki rozpoznawania przy mniejszej liczbie ograniczeń i różnych ulepszonych technik przetwarzania obrazu w tym badaniu. Na etapie określania obszaru tablicy rejestracyjnej, w celu zwiększenia wydajności systemu, oryginalny obraz jest wzmacniany przez przetwarzanie obrazu. Prostokątny obszar tablicy rejestracyjnej uzyskuje się metodami przetwarzania obrazu na podstawie krawędzi na obrazie binarnym. Wykorzystując korekcję skosu, obszar płytki uzyskany na etapie określania obszaru płytki jest przygotowywany do etapu segmentacji znaków. Znaki są oddzielone od siebie metodą rzutowania pionowego na obszar tablicy rejestracyjnej.

## Bibliografia:

1. Donald J. Norris Uczenie maszynowe na Raspberry Pi Eksperymentowanie z danymi i rozpoznawaniem obrazów
2. Valentiono Zocca, Gianmario Spacagna,Daniel Slater, Peter Roelants - Python Deep Learnin
3. Machine Learning based Number Plate Detection and Recognition

Zuhaib Ahmed Shaikh, Umair Ali Khan, Muhammad Awais Rajput and Abdul Wahid Memon <https://www.scitepress.org/Papers/2016/57502/57502.pdf>

1. Recognition of Car License Plate using Morphology <https://www.isical.ac.in/~scc/seminars/SumanMitra_ISI_CRSC_Presentation.pdf>
2. A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector <https://web.inf.ufpr.br/vri/wp-content/uploads/sites/7/2019/08/laroca2018robust.pdf>
3. Automatic number plate recognition using deep learning <https://iopscience.iop.org/article/10.1088/1757-899X/1084/1/012027/pdf>
4. A gentle introduction to OCR <https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa>
5. <https://www.tensorflow.org/lite/guide/>
6. <https://docs.opencv.org/4.x/d9/df8/tutorial_root.html>
7. https://tesseract-ocr.github.io/tessdoc/