LABORATORIUM 11

Oznakowanie linii drogowych przez OpenCV

Podstawy pracy z OpenCV

Podano części programów, które wykonują niezbędne operacje

1. **Wczytaj obraz i przekonwertuj go na obraz czarno-biały**

#Czytanie

```
# Wczytanie biblioteki.
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2 . imread ("images/indeks1.jpg")
cv2 . imshow ('Input image ', img )
cv2 . waitKey ()
cv2 . destroyAllWindows ()

```

2. **Przekonwertować obraz z kolorowego na czarno-biały**

```
gray_img = cv2 . imread ("images/indeks1.jpg", cv2.IMREAD_GRAYSCALE )
cv2.imshow ('Grayscale ', gray_img )
cv2.imwrite ("images/indeks2.jpg", gray_img )
#cv2 . waitKey ()
#cv2 . destroyAllWindows ()
```

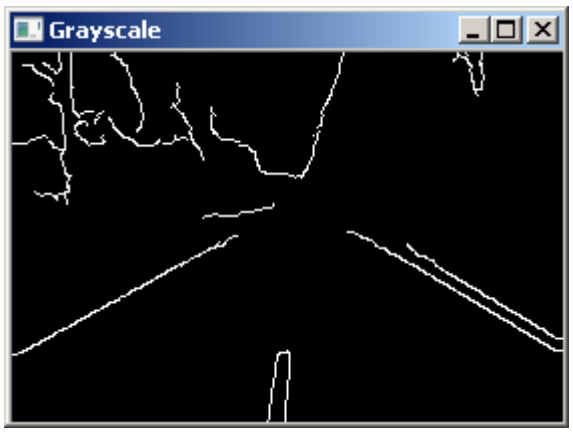
3. **Rozmycie przez filtr Gaussa**

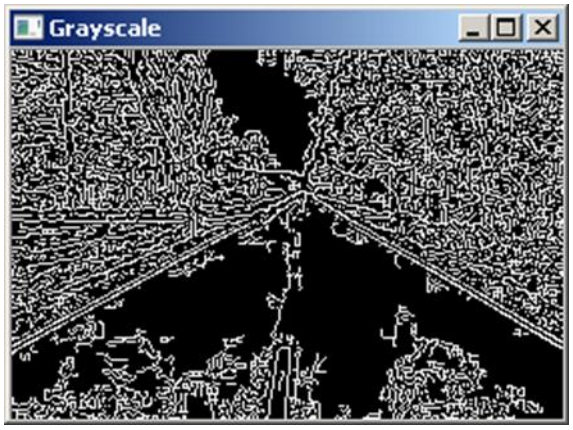
W OpenCV rozmycie jest wykonywane przez funkcję cv2.GaussianBlur

```
16    blur_kernel_size = (15,15)
17    gray_img = cv2.imread ("images/indeks2.jpg")
18    gray_blur = cv2.GaussianBlur (gray_img, blur_kernel_size, 0)
19    cv2.imshow ('Grayscale', gray_blur )
20    cv2.imwrite ("images/indeks3.jpg", gray_blur )
21    #cv2.waitKey ()
22    #cv2.destroyAllWindows ()
```

4.**Dobór konturów algorytmem Kenny'ego**

```
24    canny_low_threshold = 20
25    canny_high_threshold = 100
26    def canny (img , low_threshold , high_threshold ):
27
28        return cv2.Canny (img , low_threshold , high_threshold )
29    gray_blur = cv2 . imread ("images/indeks3.jpg")
30    blur_canny = canny ( gray_blur , canny_low_threshold , canny_high_threshold )
31    cv2.imshow ('Grayscale ', blur_canny )
32    cv2.imwrite ("images/indeks4.jpg", blur_canny )
33    #cv2.waitKey ()
34    #cv2.destroyAllWindows ()
```

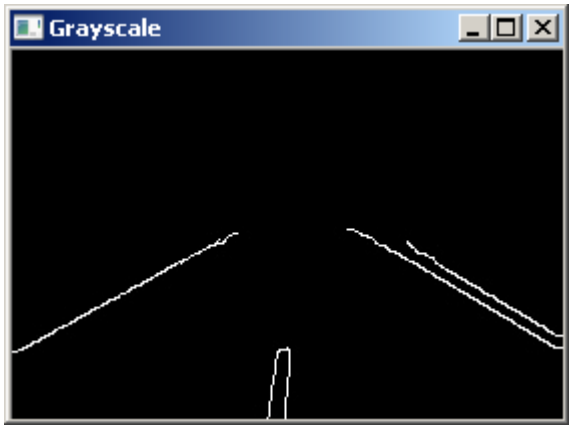




5. **Wybierz określony kontur na obrazie**

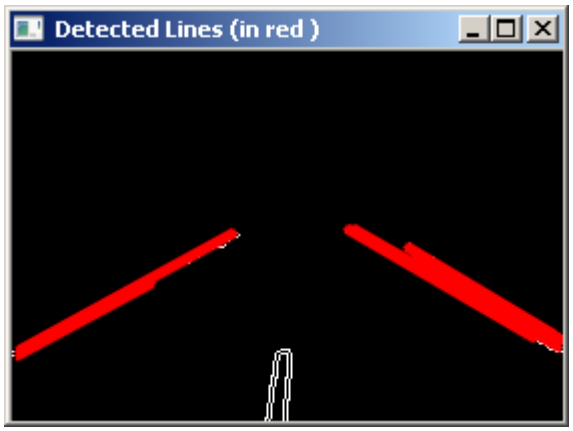
- Znajdź rozmiar obrazu
- wytnij obszar drogi, na który patrzy kamera
- utwórz czarny obraz
- przywróć rozmiar

```
39  img = cv2.imread("images/indeks4.jpg",0)
40  height, width = img.shape[:2]
41  print(img.shape[:2])
42
43  h= 97
44  w= 275
45  x= 87
46  y=0
47  img1 = img [x:x+h,y:y+w]
48  img2 =np. zeros_like ( img )
49  img2 [x:x+h,y:y+w]= img1
50  cv2.imshow ('Grayscale ', img2 )
51  cv2.imwrite ("images/indeks5.jpg", img2 )
52  cv2.waitKey ()
53  cv2.destroyAllWindows ()
```



5. **Wybór linii prostych algorytmem Huffa**

```
import math
import cv2 as cv
import numpy as np
src = cv.imread ("images/indeks5.jpg")
dst = cv.Canny (src , 50 , 200 , None , 3)
cdst = cv.cvtColor (dst , cv. COLOR_GRAY2BGR )
cdstP = np.copy ( cdst )
lines = cv.HoughLines (dst , 1, np.pi / 180 , 150 , None , 0, 0)
if lines is not None :
    for i in range (0, len ( lines )):
        rho = lines [i][0][0]
        theta = lines [i][0][1]
        a = math.cos ( theta )
        b = math.sin ( theta )
        x0 = a * rho
        y0 = b * rho
        pt1 = ( int (x0 + 1000 *(-b)), int (y0 + 1000 *(a)))
        pt2 = ( int (x0 - 1000 *(-b)), int (y0 - 1000 *(a)))
        cv.line (cdst , pt1 , pt2 , (0,0, 255 ), 3, cv. LINE_AA )
linesP = cv.HoughLinesP (dst , 1, np.pi / 180 , 50 , None , 50 , 10)
if linesP is not None :
    for i in range (0, len ( linesP )):
        l = linesP [i][0]
        cv. line (cdstP , (l[0], l[1]), (l[2], l[3]), (0,0, 255 ), 3, cv. LINE_AA )
cv.imshow (" Detected Lines (in red )", cdstP )
cv2.imwrite ("images/red.jpg", cdstP )
cv2.waitKey ()
cv2.destroyAllWindows ()
```



6. **Nakładka na oryginalny obraz**:

```
import cv2
import numpy as np
img = cv2.imread ("images/indeks1.jpg")
img1 = cv2.imread ("images/red.jpg")
img2 = cv2.addWeighted (img , 0.8, img1 , 1, 0)
cv2.imshow ('sum', img2 )
cv2.imwrite ("images/sum.jpg", img2 )
cv2.waitKey ()
cv2.destroyAllWindows ()
```

7. **Przetwarzanie plików wideo**

```
import cv2
import numpy as np
# Create a VideoCapture object and read from input file
cap = cv2.VideoCapture ("images/do.mp4")
# Check if camera opened successfully
if ( cap.isOpened ()== False ):
    print (" Error opening video file ")
# Read until video is completed
while ( cap.isOpened ()):
# Capture frame -by - frame
    ret, frame = cap.read ()
    if ret == True :
#Obraz frame trzeba obrobić, trzeba nałożyć na niego obraz Oznakowanie linii drogowych  i wydedukować!
# Display the resulting frame
        cv2.imshow ('Frame ', frame )
        #gray_image = cv2.cvtColor (frame , cv2.COLOR_BGR2GRAY )
# Display the resulting frame
# cv2.imshow (' Frame ', gray_image )
# Press Q on keyboard to exit
    if cv2.waitKey (25) & 0xFF == ord ('q'):
        break
# Break the loop
    else :
        break
# When everything done , release
# the video capture object
cap.release ()
# Closes all the frames

cv2.destroyAllWindows ()
```

Zadanie:

1. Przeczytaj dokumentację OpenCV (zawiera wszystkie niezbędne informacje o składni):

https://docs.opencv.org/3.4/d9/df8/tutorial_root.html

2. Pobierz z YouTube plik wideo z oznaczeniami dróg, na przykład:

https://www.youtube.com/watch?v=jwBaGY67olI

3. Wytnij odcinek o długości co najmniej 30 sekund

4. Wykonaj na jednej z ramek wszystkie powyższe operacje z punktów 1-6

Dla zaawansowanych (na 5!)

5. Przetwórz plik wideo z oznaczeniami dróg, aby podświetlić je na czerwono

6. Wstaw swoje imię w prawym dolnym rogu każdej ramki

7. Policz liczbę ramek w wideo

8. W trakcie obrony pracy laboratoryjnej zrozumieć i umieć wyjaśnić podstawowe operacje i algorytmy zastosowane w pracy