**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ЭТПТ**

отчет

**по лабораторной работе №4**

**по дисциплине «Машинное зрение»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6492 |  | Огурецкий Д.В. |
|  |  | Мурашко А. |
| Преподаватель |  | Моклева К.А. |

Санкт-Петербург

2019

**Задание: научиться использовть фильтры изображений.**

Ход лабораторной работы:

1. Напишите свою “легкую” реализацию cv2.threshold() только для варианта THRESH\_BINARY. Функция должна принимать значение threshold. Пусть maxVal по умолчанию всегда будет 255.

Рис.1 Исходная картинка

Рис.2 измененная картинка

Код

import cv2

import numpy as np

imggrey = cv2.imread('lab4.jpg', cv2.IMREAD\_GRAYSCALE)

def THRESH\_BINARY\_MY(img,threshold):

height,width = img.shape

for y in range(height):

for x in range(width):

if img[y][x]>threshold:

img[y][x]=255

else:

img[y][x]=0

return img

cv2.imwrite('after\_my\_tresh\_bin.jpg',THRESH\_BINARY\_MY(imggrey,127))

cv2.waitKey(0)

2. Применение на практике бинаризацию всех типов для определения разметки на дороге белого цвета:

Рис.3 ИСХОДНАЯ КАРТИНКА

THRESH\_BINARY

|  |  |
| --- | --- |
| Рис. 4.1  maxVal = 255, threshold = 127 | рис. 4.2  maxVal = 255, threshold = 170 |
| Рис.4.3  maxVal = 255, threshold = 220 |  |

Наилучшей реализацией является вариант с maxVal = 255, threshold = 170, поэтому используем даннеые параметры для применения своего метода и для дальнейших методов. Так как на первом изображении присутствует множество точек. А на последнем разметка становиться размытой.

Своя реализация THRESH\_BINARY

maxVal = 255, threshold = 170

Рис.5 СВОЯ РЕАЛИЗАЦИЯ THRESH\_BINARY

maxVal = 255, threshold = 170

Рис. 6 THRESH\_BINARY\_INV

Здесь цвета картинки в предудущем пункте инвертируется.

THRESH\_TRUNC

maxVal = 255, threshold = 170

Рис. 7 THRESH\_TRUNC

Для определения разметки метод негоден

THRESH\_TOZERO

maxVal = 255, threshold = 170

Рис. 8 THRESH\_TOZERO

THRESH\_TOZERO\_INV

maxVal = 255, threshold = 170

Рис.9 THRESH\_TOZERO\_INV

Инвертирующий метод работает плохо, так как дорога не исчезает полностью.

**Код**

**import** cv2

**import** numpy **as** np

*#Применение порогового фильтра к изображению (порог задаетсявручную):*

road1 = cv2.imread('lab4\_1.jpg', cv2.IMREAD\_GRAYSCALE)

maxVal1 = 255

threshold1 = 170

new\_threshold, road1\_1 = cv2.threshold(road1, threshold1 ,maxVal1 , cv2.THRESH\_BINARY)

cv2.imwrite(f"lab4\_1\_THRESH\_BINARY\_{threshold1}\_{maxVal1}.jpg",road1\_1)

new\_threshold, road1\_1 = cv2.threshold(road1, threshold1 ,maxVal1 , cv2.THRESH\_BINARY\_INV)

cv2.imwrite(f"lab4\_1\_THRESH\_BINARY\_INV\_{threshold1}\_{maxVal1}.jpg",road1\_1)

new\_threshold, road1\_1 = cv2.threshold(road1, threshold1 ,maxVal1 , cv2.THRESH\_TRUNC)

cv2.imwrite(f"lab4\_1\_THRESH\_TRUNC\_{threshold1}\_{maxVal1}.jpg",road1\_1)

new\_threshold, road1\_1 = cv2.threshold(road1, threshold1 ,maxVal1 , cv2.THRESH\_TOZERO)

cv2.imwrite(f"lab4\_1\_THRESH\_TOZERO\_{threshold1}\_{maxVal1}.jpg",road1\_1)

new\_threshold, road1\_1 = cv2.threshold(road1, threshold1 ,maxVal1 , cv2.THRESH\_TOZERO\_INV)

cv2.imwrite(f"lab4\_1\_THRESH\_TOZERO\_INV\_{threshold1}\_{maxVal1}.jpg",road1\_1)

**Применение порогового фильтра к изображению (порог одинаковый**

**для всех пикселей изображения, но вычисляется автоматически)**

(THRESH\_OTSU)

maxVal = 255, threshold = 150

Подобранный порог равен 150, результат почти совпадает с результатом THRESH\_BINARY

Рис. 10 THRESH\_BINARY

THRESH\_TRIANGLE

maxVal = 255, threshold = 150

Алгоритм оставляет много лишних пискелей на дороге, в отличие от OTSU

Рис.11 THRESH\_TRIANGLE

**Код**

**import** cv2

**import** numpy **as** np

*#Применение порогового фильтра к изображению (порог одинаковый*

*#для всех пикселей изображения, но вычисляется автоматически)*

road1 = cv2.imread('lab4\_1.jpg', cv2.IMREAD\_GRAYSCALE)

maxVal1 = 255

threshold1 = 100 *#127 170 220*

threshold1, road1\_1 = cv2.threshold(road1, threshold1 ,maxVal1 , cv2.THRESH\_OTSU)

cv2.imwrite(f"lab4\_1\_THRESH\_OTSU\_{threshold1}\_{maxVal1}.jpg",road1\_1)

threshold1, road1\_1 = cv2.threshold(road1, threshold1 ,maxVal1 , cv2.THRESH\_TRIANGLE)

cv2.imwrite(f"lab4\_1\_THRESH\_TRIANGLE\_{threshold1}\_{maxVal1}.jpg",road1\_1)

**Применение адаптивного вычисления порога**

Принцип работы: для каждого пикселя вычисляется взвешенное

среднее по области вокруг него размером blockSize x blockSize,

откуда вычитается константа С

Используем данные парметры функции

blocksize = 21 пиксель

C = 0.2

maxVal = 255

Метод фильтрации THRESH\_BINARY, так как он работает лучше всего для данной цели.

ADAPTIVE\_THRESH\_MEAN\_C

Рис.12 ADAPTIVE\_THRESH\_MEAN\_C

Данный метод плохо справляется с поиском разметки

ADAPTIVE\_THRESH\_GAUSSIAN\_C

Рис.13 ADAPTIVE\_THRESH\_GAUSSIAN\_C

Метод также не справляется с задачей.

Код

**import** cv2

**import** numpy **as** np

*#адаптивного вычисления порога*

road1 = cv2.imread('lab4\_1.jpg', cv2.IMREAD\_GRAYSCALE)

maxVal1 = 255

threshold1 = 100 *#127 170 220*

blocksize = 21

C = 0.2

road1\_1 = cv2.adaptiveThreshold(road1, maxVal1 , cv2.ADAPTIVE\_THRESH\_MEAN\_C ,cv2.THRESH\_BINARY , blocksize , C)

cv2.imwrite(f"lab4\_1\_ADAPTIVE\_THRESH\_MEAN\_C\_\_{threshold1}\_{maxVal1}\_{blocksize}\_{C}.jpg",road1\_1)

road1\_1 = cv2.adaptiveThreshold(road1, maxVal1 , cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C ,cv2.THRESH\_BINARY , blocksize , C)

cv2.imwrite(f"lab4\_1\_ADAPTIVE\_THRESH\_GAUSSIAN\_C\_\_{threshold1}\_{maxVal1}\_{blocksize}\_{C}.jpg",road1\_1)

Исследуем картинку ркуописного текста

Исходная картинка

Применим метод OTSU как наилучший в предыдущих пунктах

|  |  |
| --- | --- |
| рис. 14.1 | рис.14.2 |
| THRESH\_OTSU  maxVal = 255, threshold = 107 | THRESH\_TRIANGLE  maxVal = 255, threshold = 119 |

Судя по картинке первый метод лучше

Применение адаптивного вычисления порога

Используем данные парметры функции

blocksize = 21 пиксель

C = 0.2

maxVal = 255

Метод фильтрации THRESH\_BINARY, так как он работает лучше всего для данной цели.

|  |  |
| --- | --- |
| рис.15.1 | рис.15.2 |
| ADAPTIVE\_THRESH\_MEAN\_C | ADAPTIVE\_THRESH\_GAUSSIAN\_C |

Метод работает хуже , чем методы с автоматически вычисляемым порогом, применяемым ко всей картинке.

Выводы: в ходе лабораторной работы мы ознакомились с основами работы с библиотекой opencv. Работали с многомерными массивами ndarray пакета numpy для научных вычислений в python3. Научились задавать цвета с помощью RGB модели.

Метод OTSU работает лучшим образом для разных типов картинок.