|  |  |
| --- | --- |
| тов_знак_прав | МИНОБРНАУКИ РОССИИ  федеральное государственное автономное образовательное учреждение  высшего образования  «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»  **(СПбГЭТУ «ЛЭТИ»)** |

Кафедра САУ

ОТЧЕТ

по лабораторно-практической работе № 4

По курсу: «Техническое зрение»

Тема: Пороговые фильтры.

|  |  |  |
| --- | --- | --- |
| Студенты гр. 6491 |  | Квартыч Д.О. |
| Преподаватель |  | Моклева К. А. |

Санкт-Петербург

2020 г.

Цель работы: изучить принцип применения пороговых фильтров для

обработки изображений.

Основные задания:

1) Напишите свою “легкую” реализацию cv2.threshold() только для

варианта THRESH\_BINARY. Функция должна принимать значение threshold. Пусть maxVal по умолчанию всегда будет 255.

2) Примените на практике бинаризацию всех типов для следующих

изображений (нескольких каждого типа):

a) фото дорожной разметки белого цвета;

b) изображение отсканированного текстового документа;

c) фото написанного от руки или напечатанного текста.

Для каждого типа опишите, какой результат был получен.

Дополнительное задание:

В данном задании вам будет выдан набор фотографий, у которых все пиксели имеют яркость не ниже, чем некоторое значение A, и не выше, чем некоторое другое значение B. Вашей задачей будет преобразовать цвета на изображении так, чтобы гистограмма этого изображения растянулась на весь диапазон яркости [0, 255]. Заметьте, что для этого достаточно определить по изображению значения A и B, а затем диапазон яркости [A, B] пропорционально растянуть до диапазона [0, 255]. Формулу для растягивания диапазона вам предлагается вывести самостоятельно.

Основное задание №1

*Исходный код:*

1. **import** cv2
2. img = cv2.imread('E:**\S**ick Breaking**\Z**rfDctItt2g.jpg', cv2.IMREAD\_GRAYSCALE)
3. cv2.imshow('Gray', img)
4. **def** fun(img,a):
5. **for** i **in** range(len(img)):
6. **for** j **in** range(len(img[i])):
7. **if** img[i][j]>a:
8. img[i][j]=255
9. **else**:
10. img[i][j]=0
11. fun(img,110)
12. cv2.imshow('Level', img)
13. cv2.waitKey(-1)

*Результат:*

**

Основное задание №2

*Исходный код:*

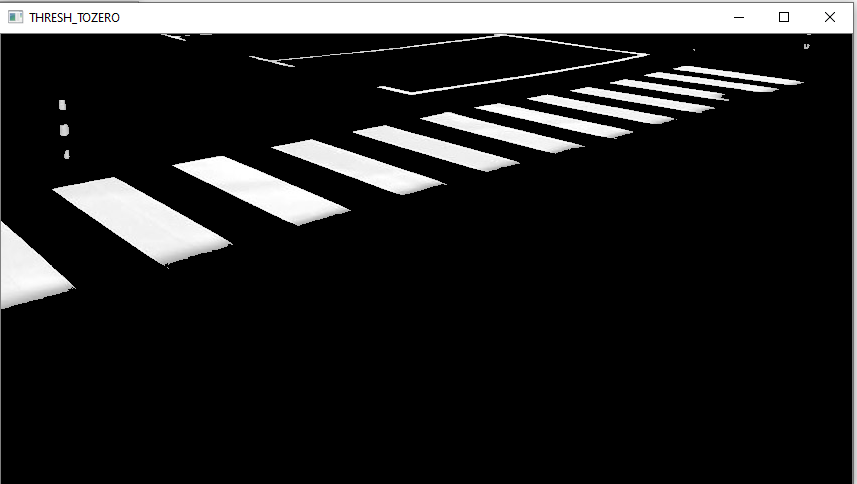
1. **import** cv2
2. **import** numpy
3. img = cv2.imread('E:**\S**ick Breaking**\d**oroga.jpg', cv2.IMREAD\_GRAYSCALE)
4. new\_threshold, new\_img = cv2.threshold(img, 150, 255, cv2.THRESH\_BINARY)
5. new\_threshold, new\_img1 = cv2.threshold(img, 150, 255, cv2.THRESH\_BINARY\_INV)
6. new\_threshold, new\_img2 = cv2.threshold(img, 230, 255, cv2.THRESH\_TRUNC)
7. new\_threshold, new\_img3 = cv2.threshold(img, 170, 255, cv2.THRESH\_TOZERO)
8. new\_img4 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY,101 ,50)
9. '''cv2.imwrite('E:**\S**ick Breaking**\c**v2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY,101 ,50).jpg',new\_img5)'''
10. cv2.imshow('THRESH\_BINARY' , new\_img)
11. cv2.imshow('THRESH\_BINARY\_INV' , new\_img1)
12. cv2.imshow('THRESH\_TRUNC' , new\_img2)
13. cv2.imshow('THRESH\_TOZERO' , new\_img3)
14. cv2.imshow('CUST' , new\_img4)
15. cv2.waitKey(-1)

Для дорог лучше подходит метод TRESH\_TOZERO, т.к. картинка монотонна, а полученное изображение четчё при ручном подборе трешхолда

Для сканов лучше подходит как адаптивный метод, так и TRESH\_TOZERO, т.к. скан документа не монотонный

Для фото написанного от руки также подходит лучше адаптивный, так как происходят резкие изменения уровня яркости на небольших расстояниях

Изображения дорожной разметки.

Изображения сканов документов.

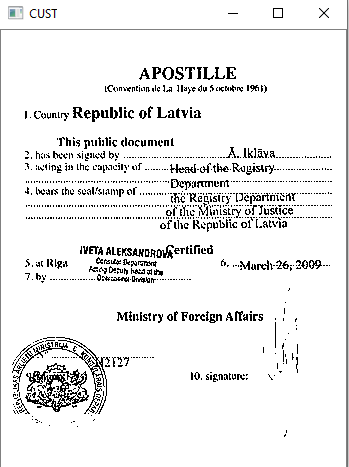
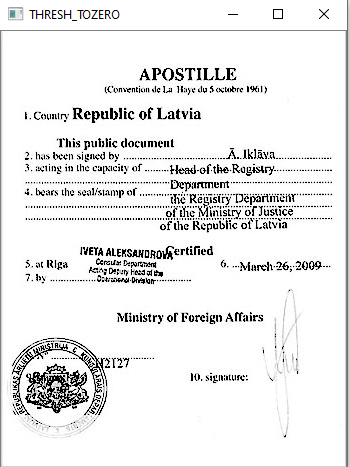
 

Фото написанного от руки текста.

|  |  |
| --- | --- |
|  |  |

Вывод: в ходе лабораторной работы было изучено применение порогового фильтра к изображению. Дополнительно были использованы ранее не изученные методы библиотек OpenCV и NumPy.