|  |  |
| --- | --- |
| 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 |  | Цуканов А.А. |
| 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:*

import cv2

import numpy

img = cv2.imread('/home/user/Documents/TSUKVA6491/zadanie/tree.jpg', cv2.IMREAD\_GRAYSCALE)

cv2.imshow('gray' , img)

def fun(img,a):

for i in range(len(img)):

for j in range(len(img[i])):

if img[i][j]>a:

img[i][j]=255

else:

img[i][j]=0

fun(img,50)

cv2.imshow('gray2' , img)

cv2.waitKey(-1)

*Результат:*





Основное задание №2

*Исходный код:*

import cv2

import numpy

img = cv2.imread('/home/user/Documents/TSUKVA6491/text2.jpg', cv2.IMREAD\_GRAYSCALE)

new\_threshold, new\_img = cv2.threshold(img, 150, 255, cv2.THRESH\_BINARY)

new\_threshold, new\_img1 = cv2.threshold(img, 150, 255, cv2.THRESH\_BINARY\_INV)

new\_threshold, new\_img2 = cv2.threshold(img, 230, 255, cv2.THRESH\_TRUNC)

new\_threshold, new\_img3 = cv2.threshold(img, 170, 255, cv2.THRESH\_TOZERO)

new\_img5 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY,101 ,50)

'''cv2.imwrite('/home/user/Documents/TSUKVA6491/cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY,101 ,50).jpg',new\_img5)'''

cv2.imshow('GRAYSCALE' , new\_img)

cv2.imshow('GRAYSCALE1' , new\_img1)

cv2.imshow('GRAYSCALE2' , new\_img2)

cv2.imshow('GRAYSCALE3' , new\_img3)

cv2.imshow('GRAYSCALE5' , new\_img5)

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\_OTSU, 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

*Результат:*

Изображения дорожной разметки.



Изображения сканов документов.

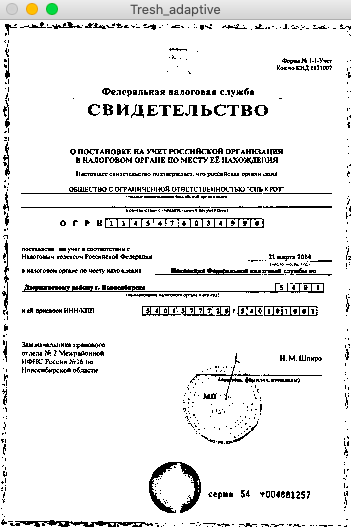


Фото написанного от руки текста.

|  |  |
| --- | --- |
| C:\Users\Artem\Desktop\cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,101 ,50).jpg | C:\Users\Artem\Desktop\cv2.threshold(img, 90, 255, cv2.THRESH_BINARY).jpg |

Дополнительное задание

*Исходный код:*

import cv2

import numpy

img = cv2.imread('/home/user/Documents/TSUKVA6491/LR\_4\_3.png', cv2.IMREAD\_GRAYSCALE)

cv2.imshow('gray' , img)

min1 = 255

max1 = 0

for i in range(len(img)):

for j in range(len(img[i])):

if min1 > img[i][j]:

min1 = img[i][j]

if max1 < img[i][j]:

max1 = img[i][j]

for i in range(len(img)):

for j in range(len(img[i])):

a = (img[i][j]-min1)\*255/(max1-min1)

img[i][j] = a

cv2.imshow('gray2', img)

cv2.waitKey(-1)

Самым интересным примером является изображение LR\_4\_3.png, так как у него смещена гистограмма относительно центра, приведем результат работы программы только для этого изображения.

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

Вывод: в ходе лабораторной работы было изучено применение порогового фильтра к изображению, также научились повышать контрастность изображения. Цвета изображения рекомендуется изменять таким образом, чтобы гистограмма пропорционально растягивалась на весь диапазон яркости. Дополнительно были использованы ранее не изученные методы библиотек OpenCV и NumPy.