МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Систем автоматического управления

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Техническое зрение»

Тема: «Пороговые преобразования»

Студент гр. 1492 Старцев Н.А.

Преподаватель Федоркова А.О.

Санкт-Петербург
2024

Цель работы: научиться применять пороговые преобразования для разделения пикселей на группы по признаку яркости

Задание 1. Используя пороговые преобразования, добиться того, чтобы птицы попали в одну группу, а фон – в другую. (изображение 2-0) Какая функция для этого подошла лучше всего? Ручной поиск, адаптивный или автоматический?

Задание 2. Используя пороговые преобразования, добиться того, чтобы фигура девушки попала в одну группу, а фон – в другую. (изображение 2-1) Какая функция для этого подошла лучше всего? Ручной поиск, адаптивный или автоматический?

Задание 3. Используя пороговые преобразования, добиться того, чтобы текст и горы попали в одну группу, а фон – в другую. (изображения 2-3 и 2-4) Какая функция для этого подошла лучше всего? Ручной поиск, адаптивный или автоматический?

Задание 4. Подключиться к камере. (Если камеры нет, попросить её у преподавателя). Используя пороговые преобразования, добиться того, чтобы Вы попали в одну группу, фон – в другую. Какая функция для этого подошла лучше всего? Ручной поиск, адаптивный или автоматический?

Задание 5. На изображении 2-2 выделить разметку.

Результаты работы

Код программы:

```
import cv2
import numpy


def nothing(x):
    pass

pathBird = "./lab2/2-0.jpg"
pathHuman = "./lab2/2-1.jpg"
pathZebra = "./lab2/2-2.jpg"
pathText = "./lab2/2-3.PNG"
pathMount = "./lab2/2-4.png"
my_img = ["Bird", "Human", "Zebra", "Text", "Mount", "Camera"]

winNameHand = "Hand"
winNameAdaptive = "Adaptive"
winNameAuto = "Auto"
winName = "Test Window"

cv2.namedWindow(winName, cv2.WINDOW_GUI_NORMAL)

imgBird = cv2.resize(cv2.imread(
    pathBird, flags=cv2.IMREAD_GRAYSCALE), (1920, 1080))
imgHuman = cv2.resize(cv2.imread(
    pathHuman, flags=cv2.IMREAD_GRAYSCALE), (1920, 1080))
imgZebra = cv2.resize(cv2.imread(
    pathZebra, flags=cv2.IMREAD_GRAYSCALE), (1920, 1080))
imgText = cv2.resize(cv2.imread(
    pathText, flags=cv2.IMREAD_GRAYSCALE), (1920, 1080))
imgMount = cv2.resize(cv2.imread(
    pathMount, flags=cv2.IMREAD_GRAYSCALE), (1920, 1080))
height, weight = imgBird.shape[0:2]
cap = cv2.VideoCapture(0)

ThresholdType = ["Hand", "Adaptive", "Auto"]
method = [cv2.THRESH_BINARY, cv2.THRESH_BINARY_INV, cv2.THRESH_TOZERO,
          cv2.THRESH_TOZERO_INV, cv2.THRESH_TRUNC, cv2.THRESH_OTSU, cv2.THRESH_TRIANGLE]
methodName = ["cv2.THRESH_BINARY", "cv2.THRESH_BINARY_INV", "cv2.THRESH_TOZERO",
              "cv2.THRESH_TOZERO_INV", "cv2.THRESH_TRUNC", "cv2.THRESH_OTSU",
"cv2.THRESH_TRIANGLE"]
methodAdaptive = [cv2.ADAPTIVE_THRESH_MEAN_C, cv2.ADAPTIVE_THRESH_GAUSSIAN_C]
methodAdaptiveName = ["cv2.ADAPTIVE_THRESH_MEAN_C",
                      "cv2.ADAPTIVE_THRESH_GAUSSIAN_C"]

cv2.createTrackbar("img", winName, 0, 5, nothing)
cv2.createTrackbar("conversion", winName, 0, 2, nothing)

flag = 0
type_Conversion = -1
index_img = -1
while (1):
    index_img = cv2.getTrackbarPos('img', winName)
    if type_Conversion != cv2.getTrackbarPos("conversion", winName):
        type_Conversion = cv2.getTrackbarPos("conversion", winName)
        flag = 1

    if flag == 1:  # нарисовать необходимые и специальные ползунки
        cv2.destroyWindow(winName)
        cv2.namedWindow(winName, cv2.WINDOW_GUI_NORMAL)
        cv2.createTrackbar("img", winName, index_img, 5, nothing)
        cv2.createTrackbar("conversion", winName, type_Conversion, 2, nothing)
        if type_Conversion == 0:  # для ручного
            cv2.createTrackbar("type", winName, 0, 4, nothing)
            cv2.createTrackbar("threshold", winName, 0, 255, nothing)
        elif type_Conversion == 1:  # для адаптивного
```

```
        # по неизвестным причинам не работают методы 2-4
        cv2.createTrackbar("type", winName, 0, 1, nothing)
        cv2.createTrackbar("type_Adaptive", winName, 0, 1, nothing)
        cv2.createTrackbar("size", winName, 0, 255, nothing)
        cv2.createTrackbar("constant", winName, 0, 255, nothing)
    else:  # для автоматического
        cv2.createTrackbar("type", winName, 0, 1, nothing)
    flag = 0

if index_img == 0:
    new_img = imgBird
elif index_img == 1:
    new_img = imgHuman
elif index_img == 2:
    new_img = imgZebra
elif index_img == 3:
    new_img = imgText
elif index_img == 4:
    new_img = imgMount
else:
    new_img = cv2.resize(cv2.cvtColor(cap.read()[1], cv2.COLOR_RGB2GRAY), (1920, 1080))

if type_Conversion == 0:
    typ = cv2.getTrackbarPos('type', winName)
    threshold = cv2.getTrackbarPos('threshold', winName)
    threshold_new, new_img = cv2.threshold(
        new_img, threshold, 255,  method[typ])
elif type_Conversion == 1:
    typ = cv2.getTrackbarPos('type', winName)
    type_Ada = cv2.getTrackbarPos('type_Adaptive', winName)
    size = cv2.getTrackbarPos('size', winName)
    c = cv2.getTrackbarPos('constant', winName)
    new_img = cv2.adaptiveThreshold(
        new_img, 255, methodAdaptive[type_Ada], method[typ], size*2+3, c)
else:
    typ = cv2.getTrackbarPos('type', winName)
    threshold_new, new_img = cv2.threshold(new_img, 19, 255, method[typ+5])

#Подписываем картинку
new_img = cv2.cvtColor(new_img, cv2.COLOR_BGR2RGB)
cv2.putText(new_img, my_img[index_img], (weight-550, height-400),
            cv2.FONT_HERSHEY_DUPLEX, fontScale=1, color=(125, 0, 255), thickness=3)
cv2.putText(new_img, ThresholdType[type_Conversion], (weight-550, height-350),
            cv2.FONT_HERSHEY_DUPLEX, fontScale=1, color=(125, 0, 255), thickness=3)
if type_Conversion == 0:
    cv2.putText(new_img, methodName[typ], (weight-550, height-300),
            cv2.FONT_HERSHEY_DUPLEX, fontScale=1, color=(125, 0, 255), thickness=3)
    cv2.putText(new_img, str(threshold_new), (weight-550, height-250),
            cv2.FONT_HERSHEY_DUPLEX, fontScale=1, color=(125, 0, 255), thickness=3)
elif type_Conversion == 1:
    cv2.putText(new_img, methodName[typ], (weight-550, height-300),
            cv2.FONT_HERSHEY_DUPLEX, fontScale=1, color=(125, 0, 255), thickness=3)
    cv2.putText(new_img, methodAdaptiveName[type_Ada], (weight-550, height-250),
            cv2.FONT_HERSHEY_DUPLEX, fontScale=1, color=(125, 0, 255), thickness=3)
    cv2.putText(new_img, str(size*2+3), (weight-550, height-200),
            cv2.FONT_HERSHEY_DUPLEX, fontScale=1, color=(125, 0, 255), thickness=3)
    cv2.putText(new_img, str(c), (weight-550, height-150),
            cv2.FONT_HERSHEY_DUPLEX, fontScale=1, color=(125, 0, 255), thickness=3)
else:
    cv2.putText(new_img, methodName[typ+5], (weight-550, height-300),
            cv2.FONT_HERSHEY_DUPLEX, fontScale=1, color=(125, 0, 255), thickness=3)

cv2.imshow(winName, new_img)
key = cv2.waitKey(100)
if key == 32:
    break
```

Примеры работы программы:
Задание 1.

Рисунок 1 – Результат ручного поиска

Рисунок 2 – Результат работы адаптивного поиска

Рисунок 3 – Результат работы автоматического поиска

Для отделения маленьких объектов на изображении лучше всего подходит адаптивный поиск.

Задание 2.

Рисунок 1 – Результат ручного поиска

Рисунок 2 – Результат работы адаптивного поиска

Рисунок 3 – Результат работы автоматического поиска

Для отделения крупных объектов на изображении лучше всего подходит ручной и автоматический поиск.

Задание 3

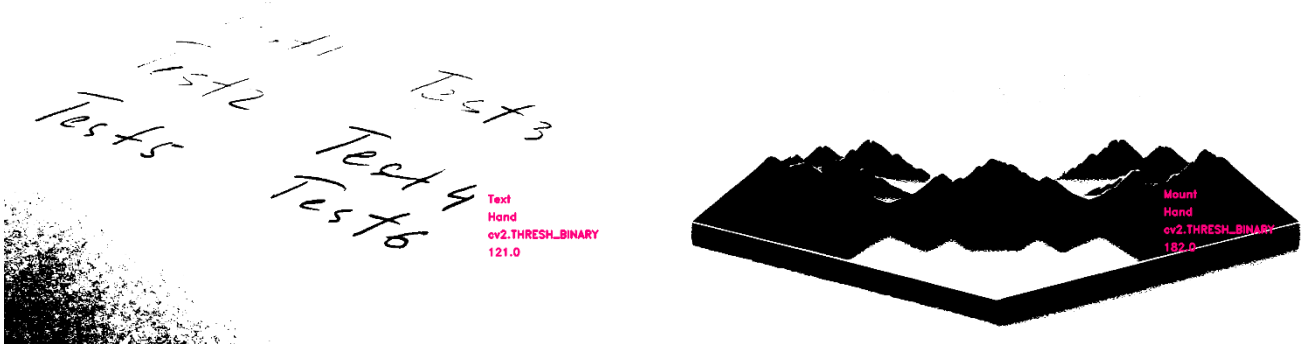

Рисунок 1 – Результат ручного поиска

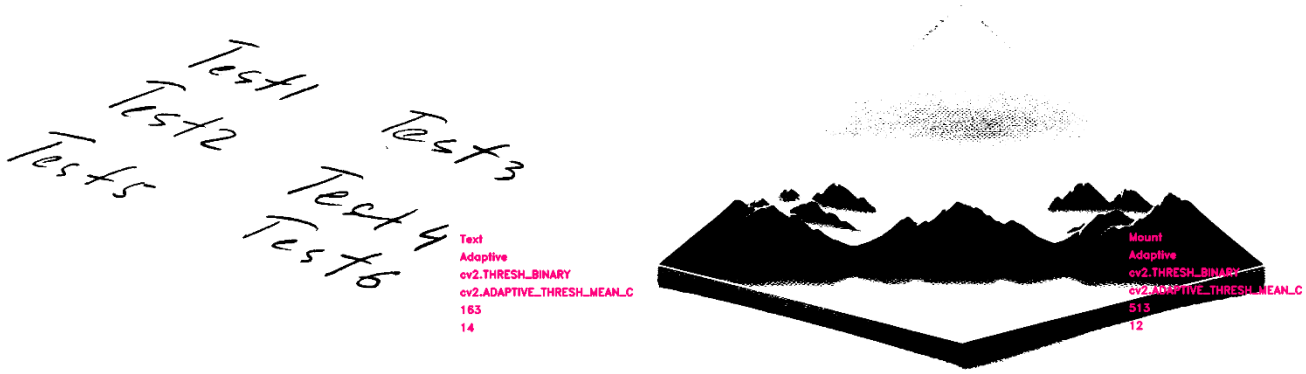

Рисунок 2 – Результат работы адаптивного поиска

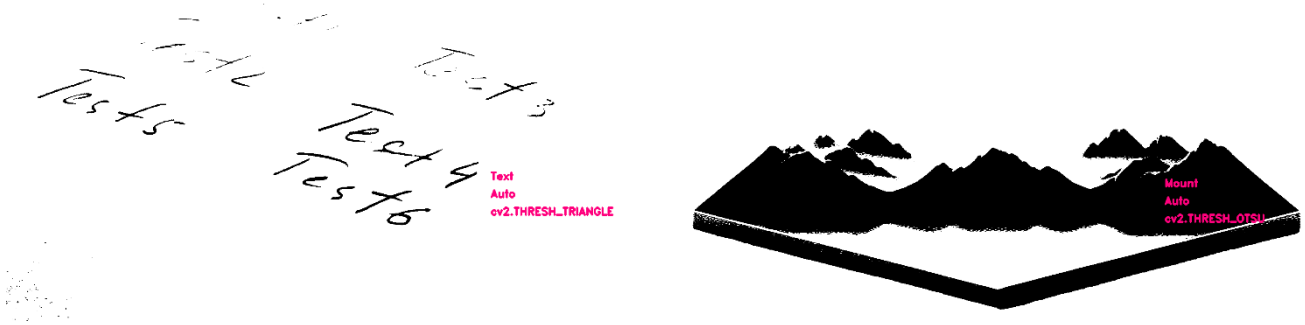

Рисунок 3 – Результат работы автоматического поиска

Для отделения текста на изображении лучше всего подходит адаптивный поиск, а для отделения гор от фона лучше подходит ручной и автоматический поиск.

Задание 4

Рисунок 1 – Результат ручного поиска

Рисунок 2 – Результат работы адаптивного поиска

Рисунок 3 – Результат работы автоматического поиска

Для отделения объектов на изображении лучше всего подходит адаптивный поиск

Задание 5

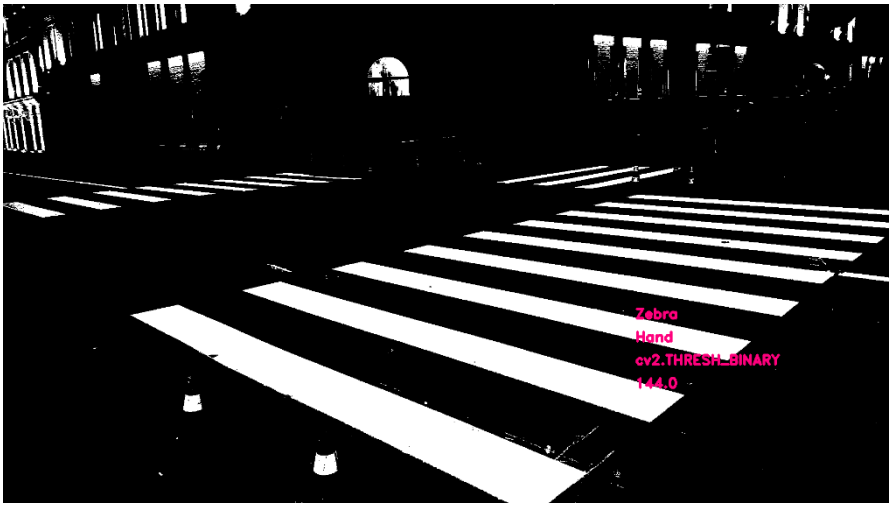

Рисунок 1 – Результат ручного поиска

Рисунок 2 – Результат работы адаптивного поиска

Рисунок 3 – Результат работы автоматического поиска

Для отделения объектов на изображении лучше всего подходит ручной и автоматический поиск.

Вывод.

В ходе лабораторной работы были изучены пороговые преобразования. Ручное преобразование, ориентируется на заданный порог, хорошо подходит для поиска крупных объектов на рисунке, когда весь объект отличается по яркости от фона. Адаптивный поиск пропускает пиксели рассматривая область вокруг пикселя, хорошо подходит для поиска малых объектов на рисунке, когда объект имеет равномерную яркость. Автоматический поиск похож на ручной, но порог яркости определяется автоматически методом отсу или методом треугольников.