МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по лабораторной работе №2 по курсу**

**«АЛГОРИТМЫ ЦИФРОВОЙ ОБРАБОТКИ МУЛЬТИМЕДИА»**

Работу выполнил

Студент 49 группы

Прокопенко Е. К

Преподаватель:

Крамаренко А. А.

Краснодар

2024

**Цель работы:** Реализовать трекинг красного объекта в камере, красный объект необходимо поднести к камере, система его находит и выделяет черным прямоугольником, далее при движении красного объекта перед камерой черный прямоугольник движется за ним.

**Ход работы:**

**Задание 1**. Прочитать изображение с камеры и перевести его в формат

HSV.

Для перевода изображения в HSV формат использовалась функция из прошлой лабораторной работы

*hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)*

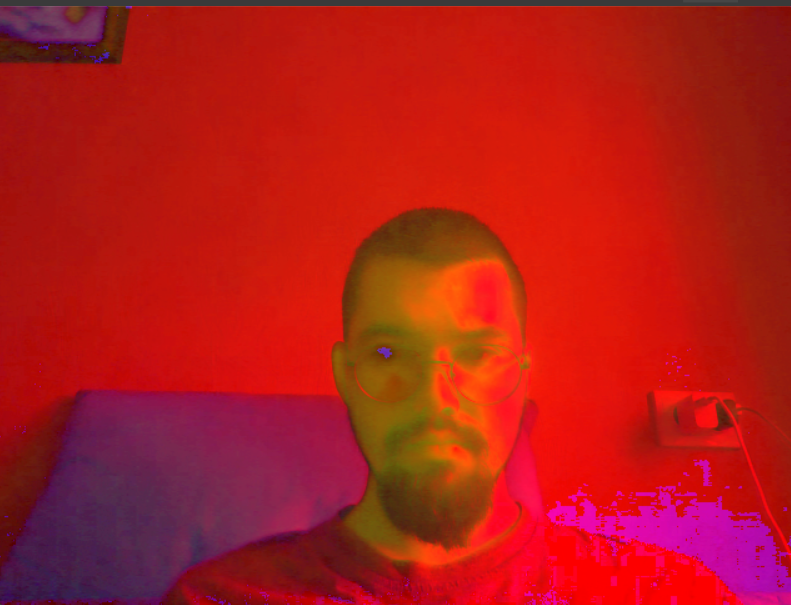


Рисунок 1 – изображение HSV.

**Задание 2.** Применить фильтрацию изображения с помощью команды

inRange и оставить только красную часть, вывести получившееся изображение

на экран(treshold) выбрать красный объект и потестировать параметры фильтрации, подобрав их нужного уровня.

Сначала был определён промежуток для красного цвета lower\_red = np.array([0, 48, 80]), upper\_red = np.array([20, 255, 255]), а затем использовалась функция cv2.inRange, которая возвращает маску, где содержится бинарное изображение (черно-белое), в котором пиксели, соответствующие заданным критериям (т.е., попадающие в диапазон между lower\_red и upper\_red), имеют значение 255, а остальные пиксели имеют значение 0. После чего, для получения итогового изображения была использована операция result = cv2.bitwise\_and(frame, frame, mask=mask).

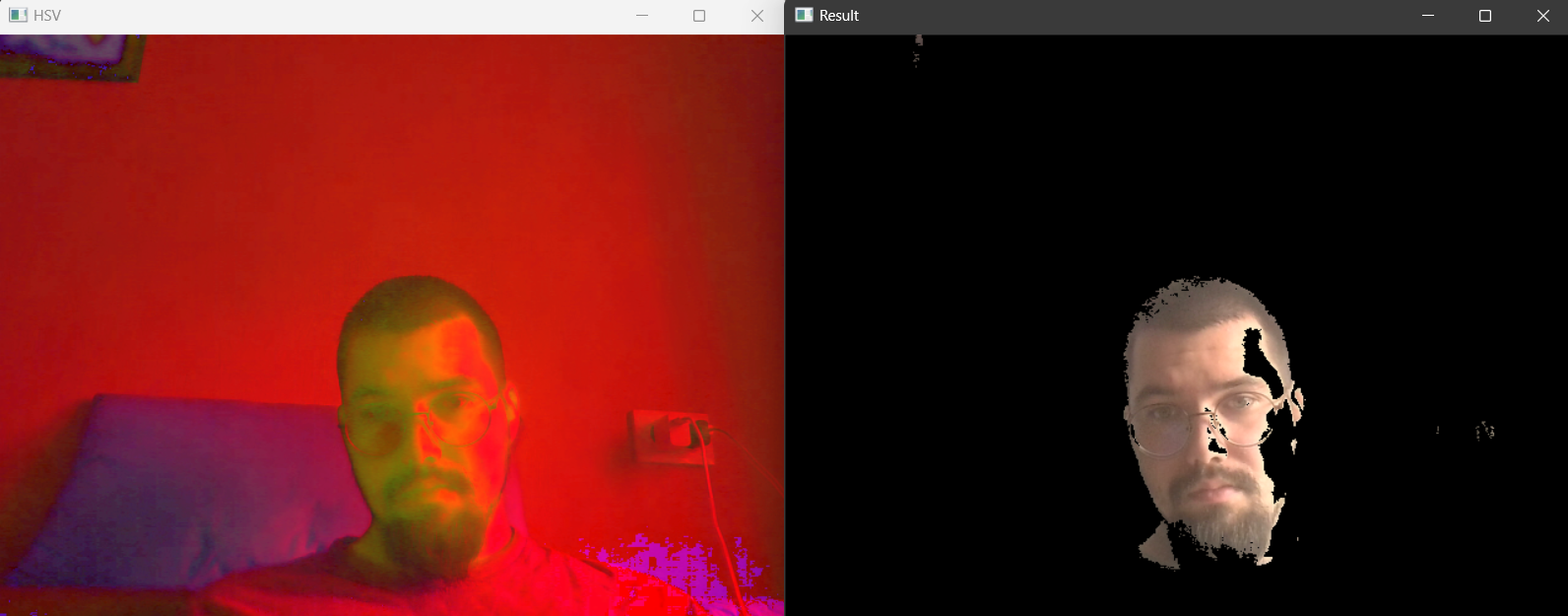


Рисунок 2 – Отфильтрованное изображение

**Задание 3.** Провести морфологические преобразования (открытие и закрытие) фильтрованного изображения, вывести результаты на экран, посмотреть смысл подобного применения операций erode и dilate.

В данном задании мы также выполняем фильтрацию изображения. После чего определяем важный компонент - структурирующий элемент (ядро). Он используется для определения формы и размера области, на которую будет применяться морфологическая операция.

В математическом смысле операции морфологической обработки выполняются на уровне битовых пикселей. Ядро скользит по изображению, и для каждой его позиции выполняются битовые логические операции, чтобы определить, какие пиксели будут активированы или деактивированы в результате операции.Далее применяются две операции:

1. Морфологическое открытие (cv2.MORPH\_OPEN) используется, чтобы удалить шумы и мелкие объекты на изображении. Он выполняет эрозию (уменьшение) изображения, чтобы убрать небольшие объекты, а затем дилатацию (расширение), чтобы восстановить оставшиеся объекты в исходных размерах.
2. Морфологическое закрытие (cv2.MORPH\_CLOSE) используется, чтобы заполнить маленькие пробелы и разрывы в объектах на изображении.

Он выполняет дилатацию для заполнения пробелов, а затем эрозию для восстановления размеров объектов.

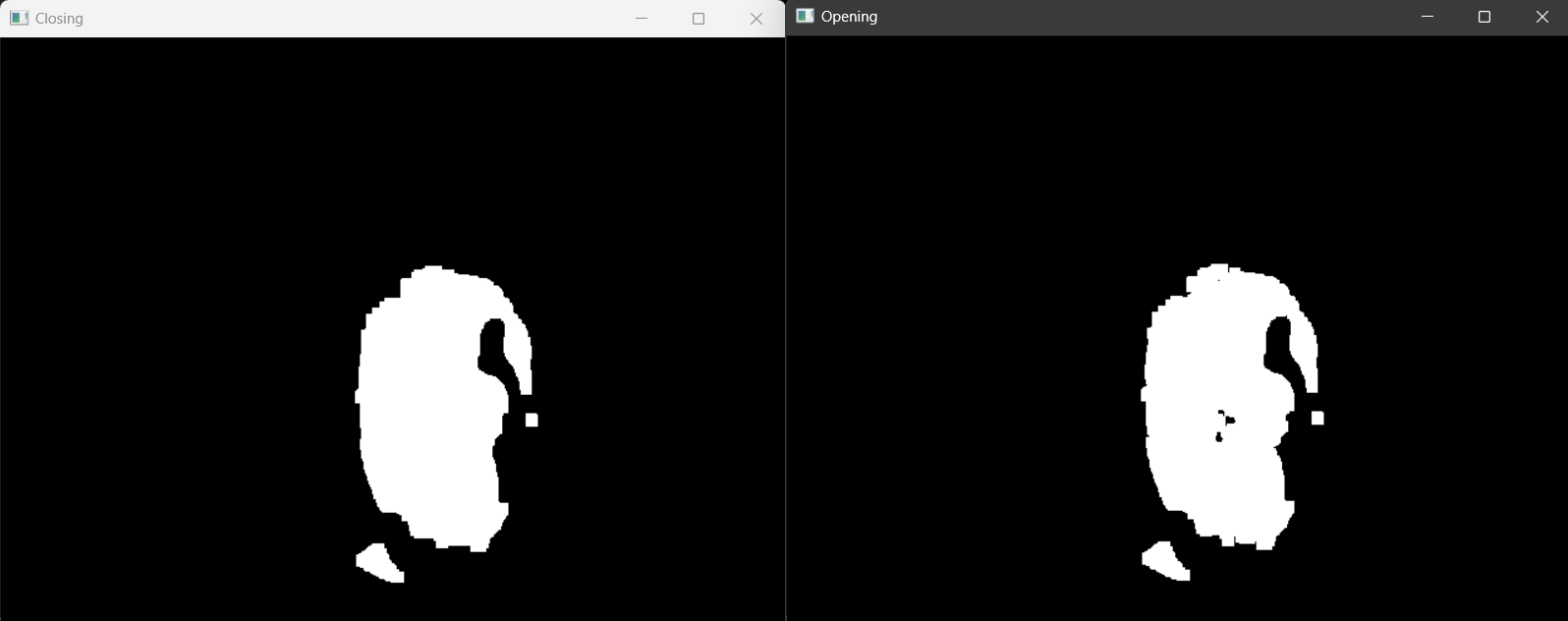


Рисунок 3 – Морфологические преобразования.

**Задание 4.** Найти моменты на полученном изображении 1 первого

порядка, найти площадь объекта.

Моменты на изображении находятся используя moments = cv2.moments(contour). Моменты изображения — это средневзвешенные значения интенсивности пикселей изображения или функция таких моментов. Они представляют характеристики объекта на изображении,

такие как площадь и центр массы.

m10 - это момент первого порядка по X (сумма произведений координаты X каждого пикселя объекта на его интенсивность),

m00 - это момент нулевого порядка (общая площадь объекта, который был выделен маской на изображении).

Используя эти значения, можно найти горизонтальную координату центра массы объекта на изображении.

**Задание 5.** На основе анализа площади объекта найти его центр и

построить черный прямоугольник вокруг объекта. Сделать так, чтобы на видео выводился полученный черный прямоугольник, причем на новом кадре новый.

Первым шагом были применены морфологические операции, описанные ранее для разделения близких объектов на изображении. После чего были найдены контуры для четкого разделения нескольких выбранных объектов.

Для построения прямоугольника были найдены координаты центра массы объекта на изображении, помощью моментов первого порядка, описанных ранее. Центр массы - это точка, в которой можно представить всю массу объекта, как если бы она была сосредоточена.

После чего рисуем прямоугольник, определяя его координаты cv2.rectangle(drawing\_frame, (x, y), (x + w, y + h), (0, 0, 0), 2)

Координаты сдвигаются относительно центра массы на значение ширины и высоты объекта влево и вверх.



Рисунок 4 – Выделение красного объекта в прямоугольник

**Листинг программ**

import cv2

import numpy as np

def main():

# Инициализация камеры

cap = cv2.VideoCapture(0)

while True:

# Чтение кадра с камеры

ret, frame = cap.read()

if not ret:

break

#Задание 1

# Преобразование в HSV

hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

#Задание 2

# Диапазон для красного цвета в HSV

lower\_red = np.array([0, 48, 80])

upper\_red = np.array([20, 255, 255])

# Создание маски для красного цвета

mask = cv2.inRange(hsv, lower\_red, upper\_red)

# Применение маски к исходному изображению

result = cv2.bitwise\_and(frame, frame, mask=mask)

#Задание 3

# Морфологические преобразования

kernel = np.ones((5, 5), np.uint8)

opening = cv2.morphologyEx(mask, cv2.MORPH\_OPEN, kernel, iterations=2)

closing = cv2.morphologyEx(opening, cv2.MORPH\_CLOSE, kernel, iterations=2)

#Задание 4-5

# Поиск контуров и вычисление моментов

contours, \_ = cv2.findContours(closing, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Создание копии исходного кадра для рисования

drawing\_frame = frame.copy()

for contour in contours:

moments = cv2.moments(contour)

area = moments['m00']

if area > 1000: # Фильтрация по площади

x, y, w, h = cv2.boundingRect(contour)

# Построение прямоугольника вокруг объекта

cv2.rectangle(drawing\_frame, (x, y), (x + w, y + h), (0, 0, 0), 2)

# Вывод информации о площади

cv2.putText(drawing\_frame, f'Area: {int(area)}', (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9,

(0, 0, 0), 2)

# Вывод результатов на несколько окон

cv2.imshow('Original', frame)

cv2.imshow('HSV', hsv)

cv2.imshow('Mask', mask)

cv2.imshow('Result', result)

cv2.imshow('Opening', opening)

cv2.imshow('Closing', closing)

cv2.imshow('Final Result', drawing\_frame)

# Ожидание нажатия клавиши 'q' для выхода из цикла

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

if \_\_name\_\_ == "\_\_main\_\_":

main()