Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №2**

**Дисциплина: Алгоритмы цифровой обработки мультимедиа**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.М. Нагалевский

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Крамаренко

Лабораторная работа №2

Реализация трекинга

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

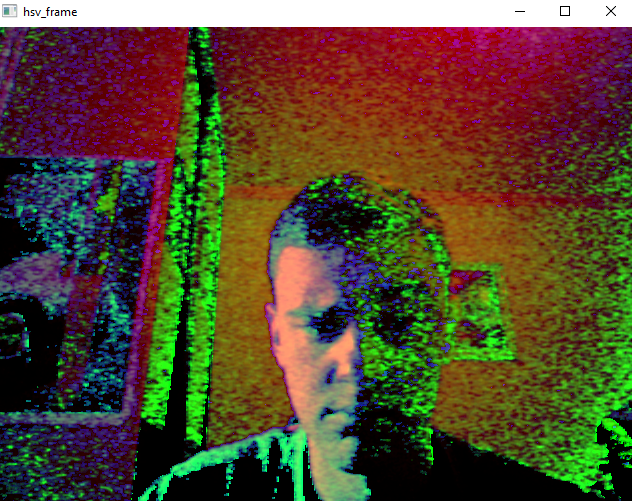


Рисунок 1 – HSV формат

**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.

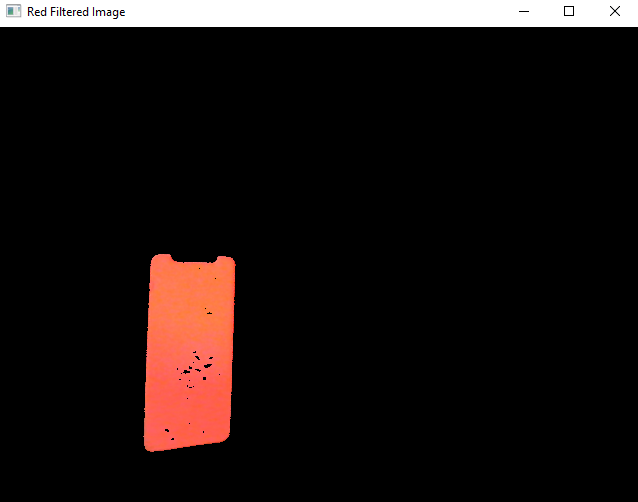


Рисунок 1 - выделение красного цвета

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

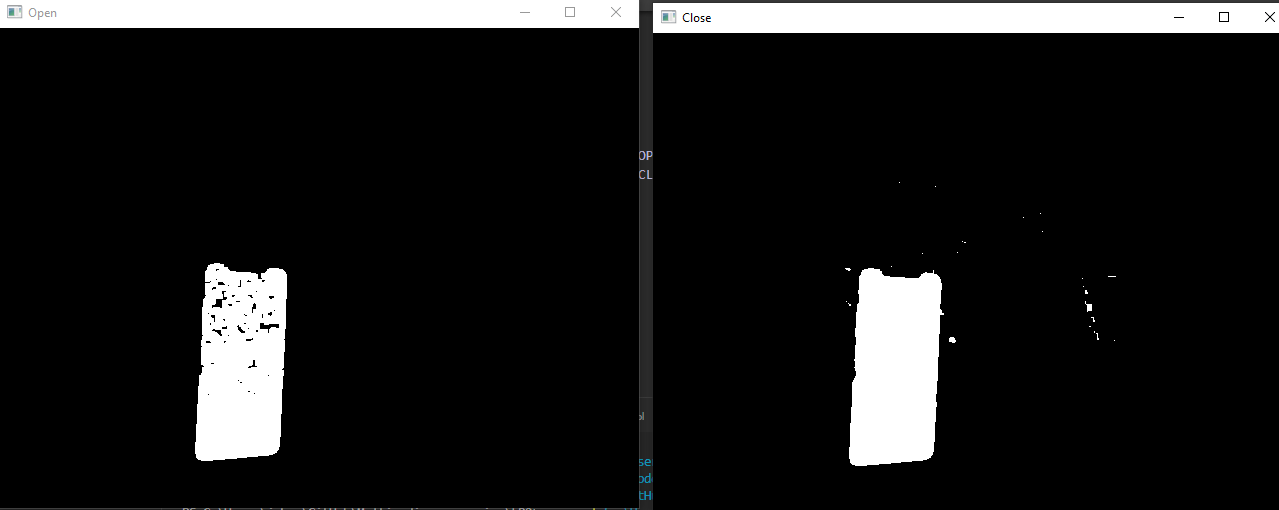


Рисунок 2: операция erode и dilate

**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.

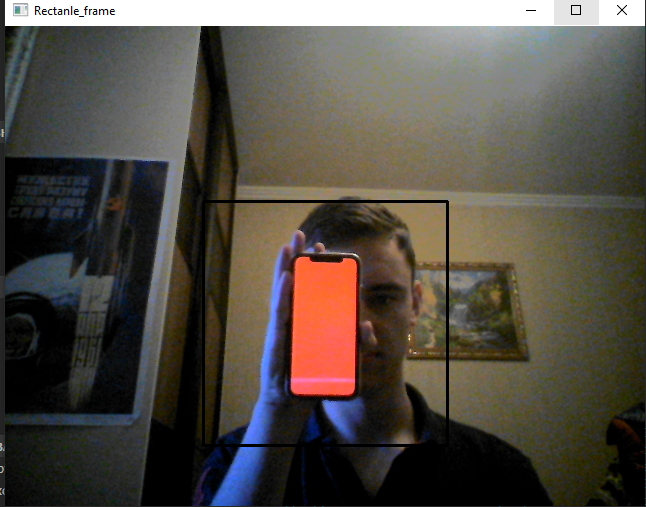


Рисунок 5: трекинг красного цвета

**Листинг программы**

Файл task1.py:

import cv2

import numpy as np

cap = cv2.VideoCapture(0)

while **True**:

    ret, frame = cap.read()

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

    cv2.imshow('hsv\_frame', hsv)

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

        break

cap.release()

cv2.destroyAllWindows()

Файл task2.py:

import cv2

import numpy as np

cap = cv2.VideoCapture(0)

while **True**:

    ret, frame = cap.read()

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

    lower\_red = np.array([0, 100, 200])

    upper\_red = np.array([10, 255, 255])

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

    red\_filtered\_frame = cv2.bitwise\_and(frame, frame, *mask* = mask)

    cv2.imshow('Red Filtered Image', red\_filtered\_frame)

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

        break

cap.release()

cv2.destroyAllWindows()

Файл task3.py:

import cv2

import numpy as np

cap = cv2.VideoCapture(0)

while **True**:

    ret, frame = cap.read()

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

    lower\_red = np.array([0, 100, 100])

    upper\_red = np.array([10, 255, 255])

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

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

    image\_opening = cv2.morphologyEx(mask, cv2.MORPH\_OPEN, kernel)

    image\_closing = cv2.morphologyEx(mask, cv2.MORPH\_CLOSE, kernel)

    cv2.imshow("Open", image\_opening)

    cv2.imshow("Close", image\_closing)

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

        break

cap.release()

cv2.destroyAllWindows()

def erode(*image*, *kernel*):

    m, n = image.shape

    km, kn = kernel.shape

    hkm = km // 2

    hkn = kn // 2

    eroded = np.copy(image)

    for i in range(hkm, m - hkm):

        for j in range(hkn, n - hkn):

            eroded[i, j] = np.min(

                image[i - hkm:i + hkm + 1, j - hkn:j + hkn + 1][kernel == 1])

    return eroded

def dilate(*image*, *kernel*):

    m, n = image.shape

    km, kn = kernel.shape

    hkm = km // 2

    hkn = kn // 2

    dilated = np.copy(image)

    for i in range(hkm, m - hkm):

        for j in range(hkn, n - hkn):

            dilated[i, j] = np.max(

                image[i - hkm:i + hkm + 1, j - hkn:j + hkn + 1][kernel == 1])

    return dilated

Файл task4.py:

import cv2

import numpy as np

cap = cv2.VideoCapture(0)

while **True**:

    ret, frame = cap.read()

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

    lower\_red = np.array([0, 100, 100])

    upper\_red = np.array([10, 255, 255])

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

    moments = cv2.moments(mask)

    area = moments['m00']

    if area > 0:

        width = height = int(np.sqrt(area))

        c\_x = int(moments["m10"] / moments["m00"])

        c\_y = int(moments["m01"] / moments["m00"])

        cv2.rectangle(frame,

            (c\_x - (width // 12), c\_y - (height // 12)),

            (c\_x + (width // 12), c\_y + (height // 12)),

            (0, 0, 0), 2)

    cv2.imshow('Rectanle\_frame', frame)

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

        break

cap.release()

cv2.destroyAllWindows()