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 выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Миленченко А.Р.

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Крамаренко А. А.

**Цель:** 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.

Было прочитано изображение с камеры и переведено его в формат HSV.

Для этого был создан цикл, в котором с помощью метода read класса VideoCapture, сначала извлекается фрейм, а затем переводится в формат hsv с помощью функции cvtColor



Рисунок 1 – Результат перевода видеопотока в hsv формат

Применяем фильтрацию изображения с помощью команды inRange и оставлена только красная часть. Было выведено получившееся изображение на экран (treshold), выбран красный объект и протестированы параметры фильтрации.

Для этого были созданы два промежутка цветов в формате hsv. Затем с помощью функции inRange были оставлены только пиксели, чей цвет попадает в данные промежутки. Полученные две маски объединены функцией bitwise\_or. Затем с помощью функции bitwise\_and были выброшены на начальном изображении пиксели, не попавшие в полученную маску.

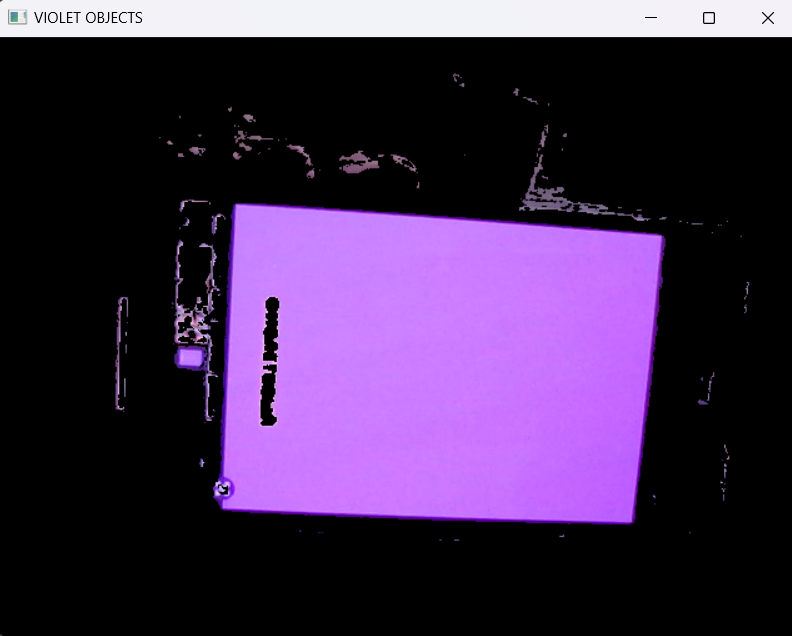


Рисунок 2 – Результат применения маски к начальному изображению

1. Были проведены морфологические преобразования (открытие и закрытие) фильтрованного изображения, выведены результаты на экран.

Для этого создается ядро для морфологических преобразований размером 5x5 с помощью пакета numpy. Затем с помощью функции cv2.morphologyEx применяются поочередно операции открытия и закрытия к маске, получаемой в предыдущем задании.

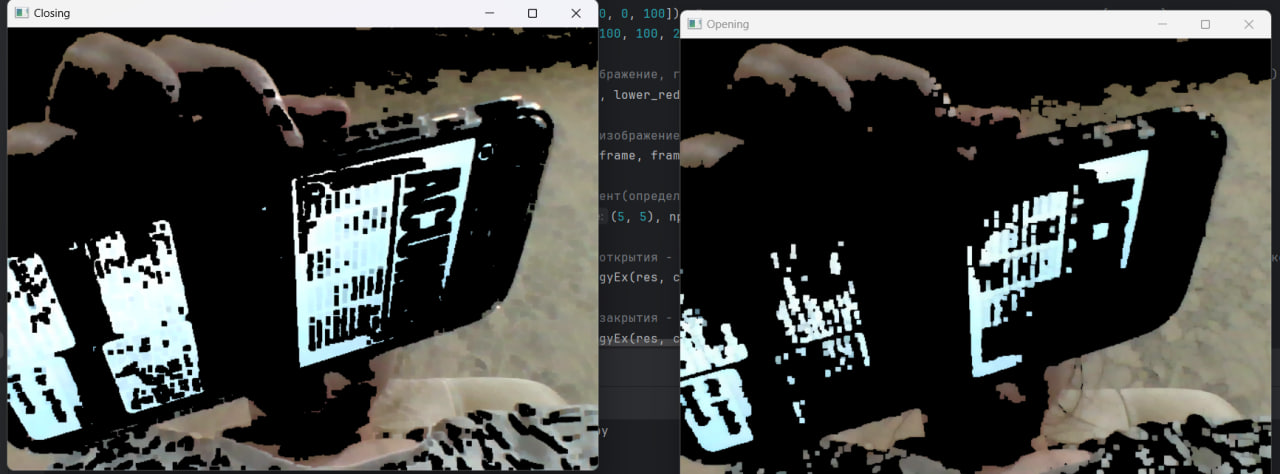
****

Рисунок 3 – Результат операции открытия и закрытия

1. На основе анализа площади объекта был найден его центр и построен черный прямоугольник вокруг объекта.

Для нахождения центра объекта моменты первого порядка делятся на момент нулевого порядка. Засчет этого получаются координаты центра найденного фиолетового объекта. Для отрисовки же прямоугольника вокруг объекта используется метод findContours(), который применяется к результату применения операции закрытия. Данный метод возвращает контуры всех найденных объектов, которые в дальнейшем используются как координаты для отрисовки нужных прямоугольников.

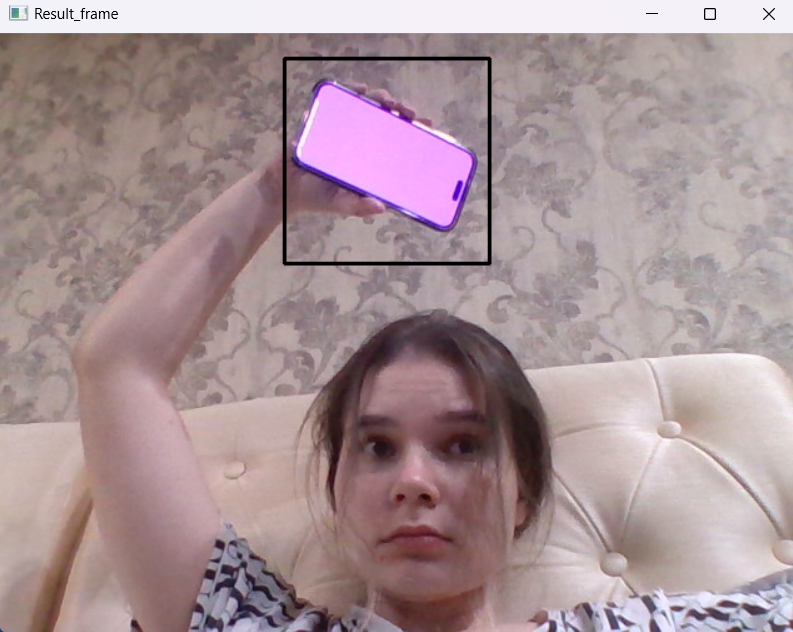
****

Рисунок 9 – Пример результат работы

**Вывод:** Был реализовать трекинг красного объекта в камере, красный объект подносится к камере, система его находит и выделяет черным прямоугольником, далее при движении красного объекта перед камерой черный прямоугольник движется за ним.

Фильтрация командой inRange

Команда inRange в OpenCV позволяет фильтровать пиксели изображения на основе диапазона значений в заданном цветовом пространстве, например, RGB или HSV. Принцип работы следующий:

1. Задается нижняя и верхняя граница диапазона значений для каждого канала цвета (например, H\_min, H\_max, S\_min, S\_max, V\_min, V\_max для HSV).
2. Для каждого пикселя изображения проверяется, попадают ли его значения в заданный диапазон для каждого канала.
3. Если значение пикселя для всех каналов находится между нижней и верхней границей, то этот пиксель сохраняется в результирующем изображении, иначе он обнуляется (становится черным).

Таким образом, команда inRange позволяет выделить объекты определенного цвета на изображении. Она часто используется в задачах компьютерного зрения, например, для сегментации объектов по цвету.

Эрозия и дилатация

Эрозия и дилатация - базовые морфологические операции в OpenCV. Они применяются для изменения формы и размера объектов на изображении. Эрозия уменьшает объекты, удаляя пиксели на границах. Она полезна для удаления шумов и отделения одного объекта от другого. Математически эрозия - это пересечение множества пикселей объекта со смещенной относительно него структурирующей функцией. Дилатация, наоборот, увеличивает объекты, добавляя пиксели на границах. Она полезна для заполнения мелких дыр и разрывов внутри объектов. Математически дилатация - это объединение множества пикселей объекта со смещенной относительно него структурирующей функцией. Часто эрозию и дилатацию применяют последовательно в виде открытия (эрозия + дилатация) и закрытия (дилатация + эрозия). Открытие позволяет удалить шумы и отделить объекты, а закрытие - заполнить дыры и разрывы.

Моменты изображения

Моменты изображения - числовые характеристики, описывающие распределение интенсивности пикселей в изображении. Различают несколько типов моментов:

* Геометрические моменты - описывают форму и размер объекта. Нулевой момент равен площади объекта, первые моменты - координатам центра масс (центроида), центральные моменты инвариантны к сдвигу.
* Физические моменты - связаны с физическими характеристиками объекта, например, массой.
* Алгебраические моменты - используются для построения инвариантных признаков объекта, например, для распознавания.

Моменты широко применяются в задачах компьютерного зрения для анализа формы объектов, вычисления их характеристик, распознавания и классификации.

Центроид объекта

Центроид (центр масс) объекта - точка, определяющая геометрический центр объекта. Она вычисляется как среднее арифметическое координат всех пикселей объекта:xc=∑i=1nxin,yc=∑i=1nyin*xc*​=*n*∑*i*=1*n*​*xi*​​,*yc*​=*n*∑*i*=1*n*​*yi*​​где $(x\_i, y\_i)$ - координаты $i$-го пикселя объекта, $n$ - число пикселей объекта.Центроид используется для определения местоположения объекта на изображении. Он может служить опорной точкой для дальнейшего анализа, например, для отслеживания движения объекта, вычисления расстояния до него и т.д.

**Ответы на вопросы:**

1. Функция inRange используется для фильтрации пикселей в пределах заданного диапазона значений в изображениях. Функция принимает три параметра: входное изображение, нижнюю границу диапазона и верхнюю границу диапазона. Пикселям, попадающим в указанный диапазон, присваивается значение белого, в то время как пикселям, выходящим за пределы диапазона, присваивается значение черного
2. Методы размывания и расширения изменяют форму и структуру объектов на изображении с использованием математических преобразований.

Стирание: метод стирания уменьшает размер объекта переднего плана и стирает границы пикселей объекта. Он удаляет пиксели на границах объекта и уменьшает изображение. Эрозия полезна для таких задач, как подавление шума и удаление мелких объектов или деталей с изображения.

Расширить: метод расширения увеличивает размер объекта переднего плана и расширяет границы пикселей объекта. Это добавляет пиксели к границам объекта и увеличивает его размер. Расширение полезно для таких задач, как заполнение отверстий, соединение разорванных частей объекта или утолщение линий на изображении.

1. Морфологическое открытие и закрытие - это сочетание эрозии и расширения, которые используются для удаления шума и мелких объектов, а также для заделки небольших отверстий или зазоров.

Opening: Морфологическое вскрытие - это сочетание эрозии с последующим расширением. Открытие помогает сгладить границы объектов и разделить связанные объекты.

Closing: Морфологическое закрытие - это сочетание расширения с последующей эрозией. Закрытие помогает завершить форму объекта и устранить небольшие пробелы.

1. Моменты изображения - это статистические показатели, которые описывают форму, структуру и распределение пикселей в изображении.

m00 - это один из моментов изображения, который называется нулевым моментом. Нулевой момент представляет собой интегральную яркость (или массу) всего изображения и используется для вычисления центра масс объекта.

area представляет собой значение нулевого момента, которое соответствует площади объекта на изображении. Это значение используется для определения площади объекта.

1. Центроид объекта на изображении - это центр масс или среднее положение всех пикселей, составляющих объект. По координатам m10 и m01 можно вычислить координаты центра масс объекта (c\_x, c\_y). Далее код рисует черный прямоугольник вокруг центра масс объекта. Пороговое значение изображения позволяет создать двоичное изображение, где пиксели объекта белые, а пиксели фона черные.

**Листинг программы**

import cv2

import numpy as np

# Задание 1: Прочитать изображение с камеры и перевести его в формат HSV

# cap = cv2.VideoCapture(0)

#

# while True:

# ret, frame = cap.read()

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

#

# cv2.imshow('HSV', hsv)

#

# if cv2.waitKey(1) & 0xFF == 27:

# break

#

# cap.release()

# cv2.destroyAllWindows()

# # Задание 2: Применить фильтрацию изображения для выделения фиолетового цвета

# # Определяем диапазоны для красного цвета

# cap = cv2.VideoCapture(0)

# while True:

# ret, frame= cap.read()

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

# violet\_mask = cv2.inRange(hsv\_image, (130, 50, 120), (160, 255, 255))

# highlighted\_violet = cv2.bitwise\_and(frame, frame, mask=violet\_mask)

# cv2.imshow('VIOLET OBJECTS', highlighted\_violet)

# if cv2.waitKey(1) & 0xFF == 27:

# break

#

# # Освобождаем ресурсы

# cap.release()

# cv2.destroyAllWindows()

#

# # # Задание 3: Морфологические преобразования

# cap = cv2.VideoCapture(0)

# while True:

# ret, frame = cap.read()

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

# # определение диапазона красного цвета в HSV

# lower\_red = np.array([0, 0, 100]) # минимальные значения оттенка, насыщенности и значения(яркости)

# upper\_red = np.array([100, 100, 255]) # максимальные значения оттенка, насыщенности и значения(яркости)

# # Маска - бинарное изображение, где пиксели, соответствующие заданному диапазону цвета, имеют значение 255 (белый), а остальные пиксели имеют значение 0 (черный).

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

# # применение маски на изображение

# res = cv2.bitwise\_and(frame, frame, mask=mask)

# # структурирующий элемент(определяет размер и форму области)

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

# # применение операции открытия - позволяет удалить шумы и мелкие объекты на изображении(удаление нежелательных пикселей или деталей)

# opening = cv2.morphologyEx(res, cv2.MORPH\_OPEN, kernel)

# # применение операции закрытия - позволяет заполнить маленькие пробелы и разрывы в объектах на изображении

# closing = cv2.morphologyEx(res, cv2.MORPH\_CLOSE, kernel)

# cv2.imshow('Opening', opening)

# cv2.imshow('Closing', closing)

# if cv2.waitKey(1) & 0xFF == 27:

# break

# cap.release()

# cv2.destroyAllWindows()

# 4- 5 задание

cap = cv2.VideoCapture(0)

previous\_cx, previous\_cy = None, None

alpha = 0.6 # коэффициент сглаживания

while True:

ret, frame = cap.read()

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

lower\_purple = np.array([130, 50, 50]) # минимальные значения оттенка, насыщенности и значения (яркости)

upper\_purple = np.array([160, 255, 255]) # максимальные значения оттенка, насыщенности и значения (яркости)

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

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

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

blurred = cv2.GaussianBlur(mask, (5, 5), 0)

moments = cv2.moments(blurred)

area = moments['m00']

if area > 500: # минимальная площадь для отсечения шумов

# ширина и высота прямоугольника равны квадратному корню из площади объекта

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

# вычисление координат центра объекта на изображении с использованием момент первого порядка

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

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

# Сглаживание движения прямоугольника

if previous\_cx is not None and previous\_cy is not None:

c\_x = int(alpha \* previous\_cx + (1 - alpha) \* c\_x)

c\_y = int(alpha \* previous\_cy + (1 - alpha) \* c\_y)

previous\_cx, previous\_cy = c\_x, c\_y

color = (0, 0, 0) # черный цвет

thickness = 2 # толщина

cv2.rectangle(frame,

(c\_x - (width // 20), c\_y - (height // 20)),

(c\_x + (width // 20), c\_y + (height // 20)),

color, thickness)

cv2.imshow('HSV\_frame', hsv)

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

if cv2.waitKey(1) & 0xFF == 27:

break

cap.release()

cv2.destroyAllWindows()

import cv2

import numpy as np

# Задание 1: Прочитать изображение с камеры и перевести его в формат HSV

# cap = cv2.VideoCapture(0)

#

# while True:

# ret, frame = cap.read()

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

#

# cv2.imshow('HSV', hsv)

#

# if cv2.waitKey(1) & 0xFF == 27:

# break

#

# cap.release()

# cv2.destroyAllWindows()

# # Задание 2: Применить фильтрацию изображения для выделения фиолетового цвета

# # Определяем диапазоны для красного цвета

# cap = cv2.VideoCapture(0)

# while True:

# ret, frame= cap.read()

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

# violet\_mask = cv2.inRange(hsv\_image, (130, 50, 120), (160, 255, 255))

# highlighted\_violet = cv2.bitwise\_and(frame, frame, mask=violet\_mask)

# cv2.imshow('VIOLET OBJECTS', highlighted\_violet)

# if cv2.waitKey(1) & 0xFF == 27:

# break

#

# # Освобождаем ресурсы

# cap.release()

# cv2.destroyAllWindows()

#

# # # Задание 3: Морфологические преобразования

# cap = cv2.VideoCapture(0)

# while True:

# ret, frame = cap.read()

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

# # определение диапазона красного цвета в HSV

# lower\_red = np.array([0, 0, 100]) # минимальные значения оттенка, насыщенности и значения(яркости)

# upper\_red = np.array([100, 100, 255]) # максимальные значения оттенка, насыщенности и значения(яркости)

# # Маска - бинарное изображение, где пиксели, соответствующие заданному диапазону цвета, имеют значение 255 (белый), а остальные пиксели имеют значение 0 (черный).

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

# # применение маски на изображение

# res = cv2.bitwise\_and(frame, frame, mask=mask)

# # структурирующий элемент(определяет размер и форму области)

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

# # применение операции открытия - позволяет удалить шумы и мелкие объекты на изображении(удаление нежелательных пикселей или деталей)

# opening = cv2.morphologyEx(res, cv2.MORPH\_OPEN, kernel)

# # применение операции закрытия - позволяет заполнить маленькие пробелы и разрывы в объектах на изображении

# closing = cv2.morphologyEx(res, cv2.MORPH\_CLOSE, kernel)

# cv2.imshow('Opening', opening)

# cv2.imshow('Closing', closing)

# if cv2.waitKey(1) & 0xFF == 27:

# break

# cap.release()

# cv2.destroyAllWindows()

# 4- 5 задание

cap = cv2.VideoCapture(0)

previous\_cx, previous\_cy = None, None

alpha = 0.6 # коэффициент сглаживания

while True:

ret, frame = cap.read()

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

lower\_purple = np.array([130, 50, 50]) # минимальные значения оттенка, насыщенности и значения (яркости)

upper\_purple = np.array([160, 255, 255]) # максимальные значения оттенка, насыщенности и значения (яркости)

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

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

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

blurred = cv2.GaussianBlur(mask, (5, 5), 0)

moments = cv2.moments(blurred)

area = moments['m00']

if area > 500: # минимальная площадь для отсечения шумов

# ширина и высота прямоугольника равны квадратному корню из площади объекта

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

# вычисление координат центра объекта на изображении с использованием момент первого порядка

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

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

# Сглаживание движения прямоугольника

if previous\_cx is not None and previous\_cy is not None:

c\_x = int(alpha \* previous\_cx + (1 - alpha) \* c\_x)

c\_y = int(alpha \* previous\_cy + (1 - alpha) \* c\_y)

previous\_cx, previous\_cy = c\_x, c\_y

color = (0, 0, 0) # черный цвет

thickness = 2 # толщина

cv2.rectangle(frame,

(c\_x - (width // 20), c\_y - (height // 20)),

(c\_x + (width // 20), c\_y + (height // 20)),

color, thickness)

cv2.imshow('HSV\_frame', hsv)

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

if cv2.waitKey(1) & 0xFF == 27:

break

cap.release()

cv2.destroyAllWindows()