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**

**Отчёт**

**по лабораторной работе №2**

**Дисциплина: АЛГОРИТМЫ ЦИФРОВОЙ ОБРАБОТКИ МУЛЬТИМЕДИА**

**Тема: «Реализация трекинга»**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К. А. Корнилов

Направление подготовки 02.03.02 Фундаментальная информатика и

информационные технологии

Направленность (профиль) Математическое и программное обеспечение

компьютерных технологий

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

**Тема:** Реализация трекинга.

**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:**

1. 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

Код:

import cv2

def trackRed():

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

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

cv2.imshow('frame', frameUpdate)

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

break

cap.release()

cv2.destroyAllWindows()

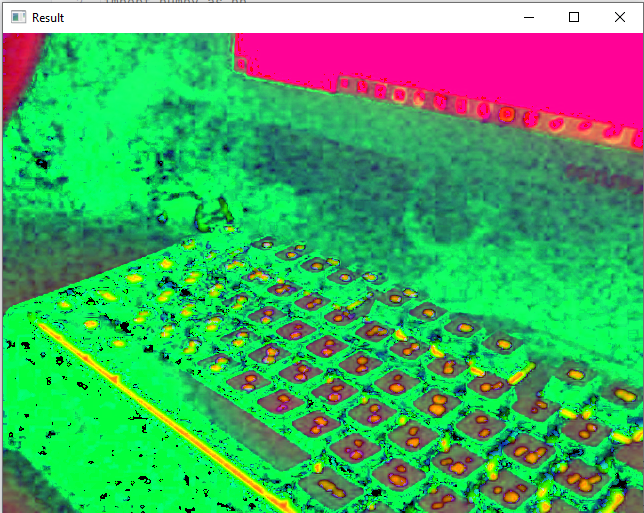


Рисунок 1 – Результат перевода видеопотока в hsv формат

1. Была применена фильтрация изображения с помощью команды inRange и оставлена только красная часть. Было выведено получившееся изображение на экран(treshold), выбран красный объект и потестированы параметры фильтрации.

Для этого были созданы два промежутка цветов в формате hsv: от (0,100,100) до (10,255,255) и от (160,100,100) до (180,255,255). 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.

Код:

def trackRed():

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

cv2.imshow("Default",frame)

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

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

# Задание 2: Создаем маску по 2 цветам

# Создаем маску

lower\_red1 = np.array([0, 100, 100])

upper\_red1 = np.array([10, 255, 255])

lower\_red2 = np.array([160, 100, 100])

upper\_red2 = np.array([180, 255, 255])

mask1 = cv2.inRange(hsv\_frame, lower\_red1, upper\_red1)

mask2 = cv2.inRange(hsv\_frame, lower\_red2, upper\_red2)

red\_mask = cv2.bitwise\_or(mask1, mask2)

# Отображаем красное на изображении

cv2.imshow('Thresholded Red', red\_mask)

res = cv2.bitwise\_and(hsv\_frame,hsv\_frame,mask=red\_mask)

cv2.imshow("Result",cv2.cvtColor(res,cv2.COLOR\_HSV2BGR))

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

break

cap.release()

cv2.destroyAllWindows()

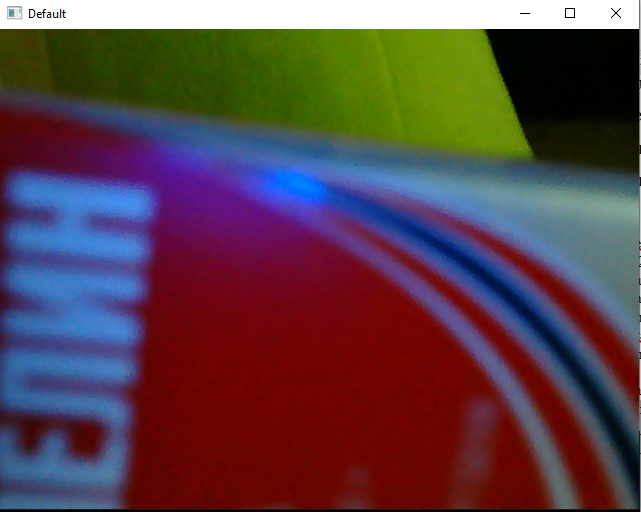


Рисунок 2 – Пример начального изображения

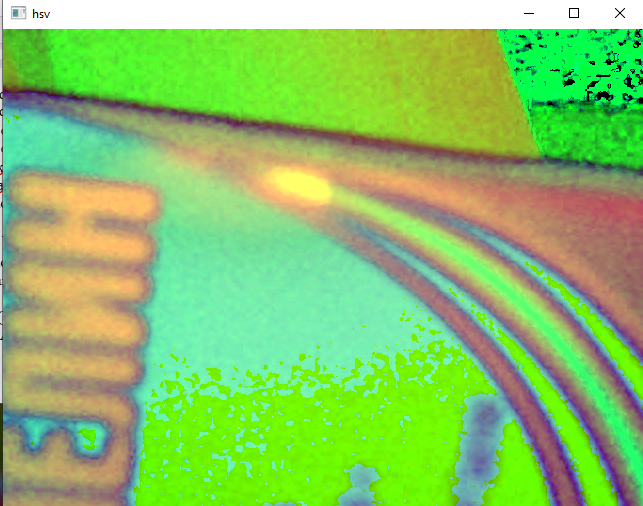


Рисунок 3 – Пример изображения в hsv



Рисунок 4 – Итоговая маска по цвету

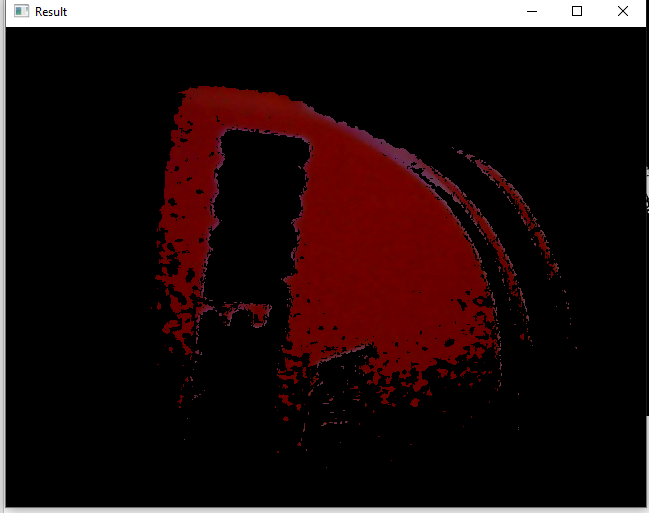


Рисунок 5 – Результат применения маски к начальному изображению

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.

Код:

def trackRed():

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

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

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

# Задание 2: Создаем маску по 2 цветам

# Создаем маску

lower\_red1 = np.array([0, 100, 100])

upper\_red1 = np.array([10, 255, 255])

lower\_red2 = np.array([160, 100, 100])

upper\_red2 = np.array([180, 255, 255])

mask1 = cv2.inRange(hsv\_frame, lower\_red1, upper\_red1)

mask2 = cv2.inRange(hsv\_frame, lower\_red2, upper\_red2)

red\_mask = cv2.bitwise\_or(mask1, mask2)

# Отображаем красное на изображении

cv2.imshow('Thresholded Red', red\_mask)

# Задание 3: Морфологические преобразования (открытие и закрытие)

kernel = np.ones((5, 5), np.uint8) # Определяем ядро для морфологических операций

opened = cv2.morphologyEx(red\_mask, cv2.MORPH\_OPEN, kernel)

closed = cv2.morphologyEx(opened, cv2.MORPH\_CLOSE, kernel) # Закрытие

# Отображаем результаты морфологических операций

cv2.imshow('Opened', opened)

cv2.imshow('Closed', closed)

res = cv2.bitwise\_and(hsv\_frame,hsv\_frame,mask=red\_mask)

cv2.imshow("Result",cv2.cvtColor(res,cv2.COLOR\_HSV2BGR))

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

break

cap.release()

cv2.destroyAllWindows()



Рисунок 6 – Результат операции открытия



Рисунок 7 – Результат операции закрытия

1. Были найдены моменты на полученном изображении 1 первого порядка, найдена площадь объекта.

Для этого используется встроенный метод moments, которые применяется к результату операции закрытия на предыдущем шаге. Значение момента 0 порядка будет являться площадью объекта. Также данный метод находит и моменты 1-ого порядка.

Код:

*# Задание 4: Нахождение моментов и площади объекта*moments = cv2.moments(closed)  
if moments['m00'] != 0:  
 area = moments['m00'] *# Площадь объекта* print(f"Area of the object: {area}")

print(f"Zero-moment:{moments['m00']}")  
 print(f"First-order moments:{moments['m01']},{moments['m10']}")

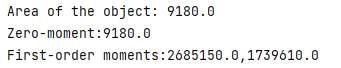


Рисунок 8 – Пример работы

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 красного 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.

Код:

moments = cv2.moments(closed)  
if moments['m00'] != 0:  
 area = moments['m00'] *# Площадь объекта* print(f"Area of the object: {area}")  
 print(f"Zero-moment:{moments['m00']}")  
 print(f"First-order moments:{moments['m01']},{moments['m10']}")  
 *# Находим центр объекта* cx = int(moments['m10'] / moments['m00'])  
 cy = int(moments['m01'] / moments['m00'])  
  
 *# Рисуем черный прямоугольник вокруг объекта* (cnts, \_) = cv2.findContours(closed.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
 for countour in cnts:  
 (x, y, w, h) = cv2.boundingRect(countour)  
 cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 0), 2)  
else:  
 area = 0  
 print("No object detected.")

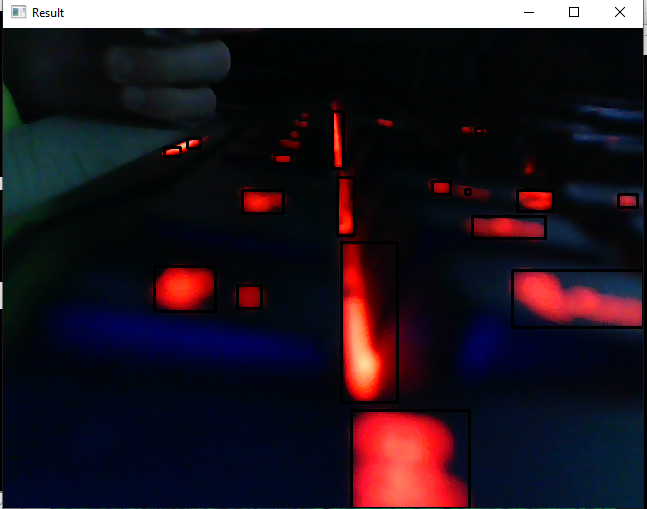


Рисунок 9 – Пример результат работы

**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.