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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_К.В.Стасюк

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

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

**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.

В данном задании сначала был определён промежуток для красного цвета, 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 (белый), а остальные пиксели имеют значение 0 (черный).После чего, для получения итогового изображения была использована операция "И" между пикселями исходного изображения (frame) и маской (mask).

Изображение выглядит как текст, Человеческое лицо, человек, улыбка

Автоматически созданное описание

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. Ядро — это набор пикселей,

где каждый пиксель имеет значение 1 или 0, и оно указывает, какие пиксели вокруг центрального пикселя будут учитываться при выполнении операции.

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.

После чего для сравнения были написаны функции Эрозии и Дилатации.

Эрозия представляет собой операцию, при которой ядро скользит по изображению, и в каждом положении проверяет, соответствуют ли пиксели внутри ядра условию.Если хотя бы один пиксель внутри ядра не соответствует условию, то центральный пиксель на выходном изображении устанавливается в 0 (черный).Это уменьшает размер объектов на изображении и удаляет мелкие детали.

Дилатация, наоборот, увеличивает размер объектов на изображении.

Ядро также скользит по изображению, и если хотя бы один пиксель внутри ядра соответствует условию,то центральный пиксель на выходном изображении устанавливается в 1 (белый).Это позволяет заполнять пробелы и увеличивать объекты на изображении.

Изображение выглядит как Человеческое лицо, человек, снимок экрана, стена

Автоматически созданное описание

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(mask). 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.

Для построения прямоугольника были найдены координаты центра массы объекта на изображении при помощи формулы Xc = m10 / m00 с помощью моментов первого порядка, описанных ранее. Центр массы - это точка, в которой можно представить всю массу объекта, как если бы она была сосредоточена.В более простых словах, это точка, в которой объект "сбалансирован" или где можно считать,что вся его масса сконцентрирована.

После чего рисуем прямоугольник, определяя его координаты как

*(c\_x - (width // 8), c\_y - (height // 8)),(c\_x + (width // 8), c\_y + (height // 8))*

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

Изображение выглядит как текст, Человеческое лицо, снимок экрана, человек

Автоматически созданное описание

Рисунок 4 – Выделение красного объекта в прямоугольник

**Листинг программ**

Файл task1­.py

import cv2  
  
# объект VideoCapture для подключения к IP-камере  
cap = cv2.VideoCapture(0)  
  
while True:  
 ret, frame = cap.read()  
  
 # преобразование изображения в формат HSV  
 hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)  
  
 cv2.imshow("HSV\_image", hsv)  
  
 # сохранение изображения  
 cv2.imwrite("task\_1.png", hsv)  
  
 # нажатие клавиши esc для выхода из цикла  
 if cv2.waitKey(1) & 0xFF == 27:  
 break  
  
# освобождение ресурсов окна  
cap.release()  
cv2.destroyAllWindows()

Файл task2.py

import cv2  
import numpy as np  
  
# объект VideoCapture для подключения к IP-камере  
cap = cv2.VideoCapture(0)  
  
while True:  
 ret, frame = cap.read()  
 # преобразование кадра в формат HSV  
 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 яркости  
 upper\_red = np.array([60, 255, 255]) # максимальные значения оттенка, насыщенности и 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)  
  
 # выполняет операцию "И" между пикселями исходного изображения (frame) и маской (mask).  
 res = cv2.bitwise\_and(frame, frame, mask=mask)  
  
 # cv2.imshow('Original\_video', frame)  
 cv2.imshow('HSV\_video', hsv)  
 cv2.imshow('Result\_video', res)  
  
 # нажатие клавиши esc для выхода из цикла  
 if cv2.waitKey(1) & 0xFF == 27:  
 break  
  
# освобождение ресурсов окна  
cap.release()  
cv2.destroyAllWindows()

Файл task3.py

import cv2  
import numpy as np  
  
  
# Операция эрозии - уменьшение изображения для удаления мелких деталей и шумов  
def erode(image, kernel):  
 m, n, \_ = image.shape # Получаем высоту и ширину изображения  
 km, kn = kernel.shape  
 hkm = km // 2  
 hkn = kn // 2  
 eroded = np.copy(image)  
  
 # Проходимся по каждому пикселю изображения, начиная с пикселей, где размер ядра помещается полностью  
 for i in range(hkm, m - hkm):  
 for j in range(hkn, n - hkn):  
 # Вычисляем минимум среди пикселей внутри ядра, только если соответствующий элемент ядра равен 1  
 eroded[i, j] = np.min(  
 #создания подматрицы (подобласти) изображения вокруг текущего пикселя  
 image[i - hkm:i + hkm + 1, j - hkn:j + hkn + 1][kernel == 1])  
  
 return eroded  
  
# Операция дилатации - увеличение изображения для заполнения недостающих частей  
def dilate(image, kernel):  
 m, n, \_ = image.shape # Получаем высоту и ширину изображения  
 km, kn = kernel.shape  
 hkm = km // 2  
 hkn = kn // 2  
 dilated = np.copy(image)  
  
 # Проходимся по каждому пикселю изображения, начиная с пикселей, где размер ядра помещается полностью  
 for i in range(hkm, m - hkm):  
 for j in range(hkn, n - hkn):  
 # Вычисляем максимум среди пикселей внутри ядра, только если соответствующий элемент ядра равен 1  
 dilated[i, j] = np.max(  
 image[i - hkm:i + hkm + 1, j - hkn:j + hkn + 1][kernel == 1])  
  
 return dilated  
  
  
  
# объект VideoCapture для подключения к IP-камере  
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 яркости  
 upper\_red = np.array([60, 255, 255]) # максимальные значения оттенка, насыщенности и 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 (ядро) размером 5x5, который представляет собой матрицу, где все элементы установлены в 1  
 # (определяет размер и форму области)  
 kernel = np.ones((5, 5), np.uint8)  
  
 # Операция открытия - 1)эрозия, 2)дилатация.  
 opening = cv2.morphologyEx(res, cv2.MORPH\_OPEN, kernel)  
  
 # Операция закрытия-1)дилатации, 2) операция эрозии.  
 closing = cv2.morphologyEx(res, cv2.MORPH\_CLOSE, kernel)  
  
 cv2.imshow('Opening', opening)  
 cv2.imshow('Closing', closing)  
  
 # Применение операции эрозии  
 # eroded = erode(res, kernel)  
 #  
 # # Применение операции дилатации  
 # dilated = dilate(res, kernel)  
 #  
 # # Отображение изображений с применением эрозии и дилатации  
 # cv2.imshow('Erosion', eroded)  
 # cv2.imshow('Dilation', dilated)  
  
  
 # нажатие клавиши esc для выхода из цикла  
 if cv2.waitKey(1) & 0xFF == 27:  
 break  
  
# освобождение ресурсов окна  
cap.release()  
cv2.destroyAllWindows()

Файл task4-5.py

import cv2

import numpy as np

# объект VideoCapture для подключения к IP-камере

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([10, 255, 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)

# вычисление моментов на основе маски(характеристик)

moments = cv2.moments(mask)

# поиск момента первого порядка- общая площадь объекта, который был выделен маской на изображении

area = moments['m00']

if area > 0:

# ширина и высота прямоугольника равны квадратному корню из площади объекта

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

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

#горизонтальная координата

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

#вертикальная

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

# отрисовка прямоугольника

color = (0, 0, 0) # черный цвет

thickness = 4 # толщина

cv2.rectangle(frame,

#верхний левый угол

(c\_x - (width // 10), c\_y - (height // 10)),

#нижний правый угол

(c\_x + (width // 10), c\_y + (height // 10)),

color, thickness)

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

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

# нажатие клавиши esc для выхода из цикла

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

break

print("Площадь объекта:", area)

# освобождение ресурсов окна

cap.release()

cv2.destroyAllWindows()