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 №5**

**Дисциплина: Алгоритмы цифровой обработки мультимедиа**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Пинский Д.А.

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

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

**Цель работы:** В рамках данной лабораторной работы будет рассматриваться решение задачи детекции движения на видео.

В рамках данной лабораторной работы будет рассматриваться решение задачи детекции движения на видео.

Решение задачи будем выстраивать исходя из следующего алгоритма:

- начать чтение из файла, прочитать первый кадр, перевести в чернобелый цвет и применить размытие Гаусса;

- подготовить файл для записи;

- далее начать цикл, который завершиться по завершению файла, внутри этого цикла:

- скопировать старый кадр;

- прочитать новый кадр, перевести в чернобелый цвет, применить размытие Гаусса;

- если чтение неуспешно, остановить цикл; - найти разницу между двумя кадрами в отдельный фрейм (frame\_diff)

- cv2.absdiff; - провести операцию двоичного разделения для фрейма (frame\_diff)

- cv2.threshold; - найти контуры объектов для фрейма (frame\_diff) - cv2.findContours;

- пройтись по контурам объектов для фрейма (frame\_diff) и найти контур площадью большей, чем наперед заданный параметр

- cv2.contourArea;

- если такой контур найден, значит было движение, записать кадр в файл;

- отобразить видео.

Ответы на контрольные вопросы.

1.

- начать чтение из файла, прочитать первый кадр, перевести в чернобелый цвет и применить размытие Гаусса;

- подготовить файл для записи;

- далее начать цикл, который завершиться по завершению файла, внутри этого цикла:

- скопировать старый кадр;

- прочитать новый кадр, перевести в чернобелый цвет, применить размытие Гаусса;

- если чтение неуспешно, остановить цикл; - найти разницу между двумя кадрами в отдельный фрейм (frame\_diff)

- cv2.absdiff; - провести операцию двоичного разделения для фрейма (frame\_diff)

- cv2.threshold; - найти контуры объектов для фрейма (frame\_diff) - cv2.findContours;

- пройтись по контурам объектов для фрейма (frame\_diff) и найти контур площадью большей, чем наперед заданный параметр

- cv2.contourArea;

- если такой контур найден, значит было движение, записать кадр в файл;

- отобразить видео.

2.

cv2.absdiff

Функция cv2.absdiff в OpenCV используется для вычисления абсолютной разницы между двумя изображениями (или массивами). Она выполняет поэлементное вычисление абсолютной разницы, что означает, что для каждого пикселя в двух входных изображениях вычисляется разница, и если она отрицательная, то результат становится положительным. Это позволяет избежать проблем с отрицательными значениями, которые могут возникнуть при обычном вычитании.

Сигнатура функции:

cv2.absdiff(src1, src2, dst)

src1: Первое входное изображение.

src2: Второе входное изображение.

dst: Выходное изображение, в котором будет храниться результат.

cv2.threshold

Функция cv2.threshold в OpenCV используется для применения порогового преобразования к изображению, что позволяет преобразовать его в бинарное изображение. Это полезно для выделения объектов на изображении, основанного на значениях яркости пикселей.

Сигнатура функции:

retval, dst = cv2.threshold(src, thresholdValue, maxValue, thresholdType)

src: Исходное изображение (должно быть в градациях серого).

thresholdValue: Пороговое значение. Пиксели с яркостью выше этого значения будут изменены в зависимости от типа порогового преобразования.

maxValue: Максимальное значение, которое будет присвоено пикселям, превышающим порог.

thresholdType: Тип порогового преобразования (флаги), определяющий, как будет выполнено преобразование.

Возвращаемые значения

retval: Используемое пороговое значение (может быть полезно для динамического порогового преобразования).

dst: Изображение после применения порогового преобразования.

cv2.findContours

Функция cv2.findContours в OpenCV используется для обнаружения контуров объектов на бинарном изображении. Контуры представляют собой кривые, соединяющие все непрерывные точки вдоль границы, имеющие одинаковый цвет или интенсивность. Эта функция полезна для анализа форм и распознавания объектов.

Сигнатура функции:

contours, hierarchy = cv2.findContours(image, mode, method)

image: Исходное изображение (должно быть бинарным, то есть иметь только два цвета: черный и белый).

mode: Режим извлечения контуров.

method: Метод аппроксимации контуров.

Возвращаемые значения

contours: Список контуров, где каждый контур представлен как массив координат (x, y) точек границы объекта.

hierarchy: (опционально) Массив, описывающий иерархию контуров (например, вложенные контуры).

cv2.contourArea

Функция cv2.contourArea в OpenCV используется для вычисления площади контура, который был обнаружен на изображении. Эта функция принимает контур в виде массива точек и возвращает площадь, заключенную внутри этого контура.

Сигнатура функции

area = cv2.contourArea(contour)

Параметры:

contour: Массив точек, представляющий контур. Этот массив должен быть замкнутым (то есть первая и последняя точки должны совпадать).

Возвращаемое значение

area: Площадь, заключенная внутри контура, выраженная в пикселях.

**Листинг программы**

import cv2

def main(path\_to\_file, kernel\_size, standart\_deviation, treshLow, min\_area):

    video = cv2.VideoCapture(path\_to\_file, cv2.CAP\_ANY)

    ret, frame = video.read()

    width = int(video.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

    height = int(video.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

    fourcc = cv2.VideoWriter.fourcc(\*'mp4v')

    outputVideo = cv2.VideoWriter("new\_test\_video.mp4", fourcc, 25, (width, height))

    ret, frame = video.read()

    gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    oldFrame = cv2.GaussianBlur(gray, (kernel\_size, kernel\_size), standart\_deviation)

    while True:

        ret, frame = video.read()

        if not ret:

            break

        gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

        blurGray = cv2.GaussianBlur(gray, (kernel\_size, kernel\_size), standart\_deviation)

        frame\_diff = cv2.absdiff(oldFrame, blurGray)

        thresh = cv2.threshold(frame\_diff, treshLow, 255, cv2.THRESH\_BINARY)[1]

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

        try:

            contour = contours[0]

            area = cv2.contourArea(contour)

            if area > min\_area:

                outputVideo.write(frame)

        except:

            pass

        oldFrame = blurGray

    outputVideo.release()

main("C:\ACOM\LR5\ЛР4\_main\_video.mov", 3, 60, 120, 20)