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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.М. Нагалевский

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

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

**Тема работы:** Выявление движения на видео.

**Ход работы:**

1. начать чтение из файла, прочитать первый кадр, перевести в черно-белый цвет и применить размытие Гаусса;
2. подготовить файл для записи;
3. далее начать цикл, который завершиться по завершению файла, внутри этого цикла:

* скопировать старый кадр;
* прочитать новый кадр, перевести в черно-белый 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.

Сначала подготовим кадр для дальнейшей обработки, а именно переведем в черно-белое представление и применим Гауссовское размытие. (рисунок 1)

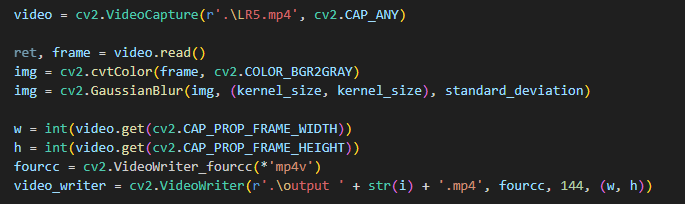


Рисунок 1 – подготовка кадра к обработке

Далее запустим цикл, где будут происходить все преобразования. Вычислим разницу между искомым изображением и новым с помощью функции cv2.absdiff(). Бинаризируем её, используя функцию cv2.threshold(), превращая пиксели, превышающие порог delta\_tresh, в белый цвет, а остальные - в черные (сохраняем только пороговое значение). Дальше найдем контуры (cv2.findContours()), используя пороговое значение. Все эти манипуляции изображены на рисунке 2.

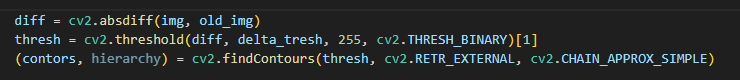


Рисунок 2 – Разница, бинаризация и нахождение контуров

И наконец, будем проходится по каждому контуру и сравнить его площадь с общей площадью кадра. Если она больше или равна значению min\_area, то записываем текущий кадр в выходное видео. Если же нет, то продолжаем проход.

Были проведены тесты, которые показали более оптимальные значения параметров для выбранного видео, а именно:

1. Размер ядра = 11
2. Стандартное отклонение = 70
3. Пороговое значение = 60
4. Минимальная площадь = 20

На рисунке 3 показан результат работы алгоритма при самых оптимальных значениях.



Рисунок 3 - Результат работы алгоритма при самых оптимальных значениях.

Также тестирование было проведено со значениями:

1. ядра 3, отклонением 50, пороговым значением 60 и площадью 20 (рисунок 4).



Рисунок 4 - Результат работы алгоритма при значениях 1.

1. ядра 3, отклонением 50, пороговым значением 20 и площадью 20 (рисунок 5).



Рисунок 5 - Результат работы алгоритма при значениях 2.

1. ядра 3, отклонением 50, пороговым значением 60 и площадью 10 (рисунок 6).



Рисунок 6 - Результат работы алгоритма при значениях 3.

**Листинг программ**

Файл main.py

import cv2

import numpy as np

i = 0

def main(kernel\_size, standard\_deviation, delta\_tresh, min\_area):

    global i

    i += 1

    video = cv2.VideoCapture(r'.\LR5.mp4', cv2.CAP\_ANY)

    ret, frame = video.read()

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

    img = cv2.GaussianBlur(img, (kernel\_size, kernel\_size), standard\_deviation)

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

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

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

    video\_writer = cv2.VideoWriter(r'.\output ' + str(i) + '.mp4', fourcc, 144, (w, h))

    while True:

        old\_img = img.copy()

        ok, frame = video.read()

        if not ok:

            break

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

        img = cv2.GaussianBlur(img, (kernel\_size, kernel\_size), standard\_deviation)

        diff = cv2.absdiff(img, old\_img)

        thresh = cv2.threshold(diff, delta\_tresh, 255, cv2.THRESH\_BINARY)[1]

        (contors, hierarchy) = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

        for contr in contors:

            area = cv2.contourArea(contr)

            if area < min\_area:

                continue

            video\_writer.write(frame)

    video\_writer.release()

kernel\_size = 3

standard\_deviation = 50

delta\_tresh = 60

min\_area = 20

main(kernel\_size, standard\_deviation, delta\_tresh, min\_area)

kernel\_size = 11

standard\_deviation = 70

delta\_tresh = 60

min\_area = 20

main(kernel\_size, standard\_deviation, delta\_tresh, min\_area)

kernel\_size = 3

standard\_deviation = 50

delta\_tresh = 20

min\_area = 20

main(kernel\_size, standard\_deviation, delta\_tresh, min\_area)

kernel\_size = 3

standard\_deviation = 50

delta\_tresh = 60

min\_area = 10

main(kernel\_size, standard\_deviation, delta\_tresh, min\_area)