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.А. Иванищев

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, а также создание переменной, которая создаёт выходное видео. (рисунок 1)

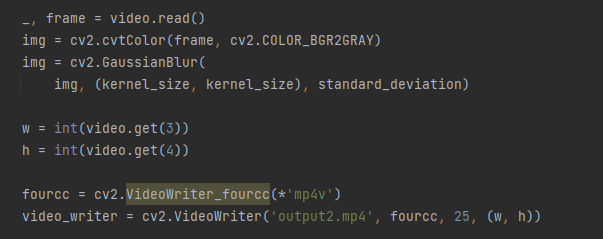


Рисунок 1 – подготовка кадра к обработке

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.

И наконец, будем проходиться по каждому контуру и сравнивать 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 = 3
2. Стандартное отклонение = 50
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. Размер ядра - 7, отклонение - 60, пороговое значение - 20 и площадью 30 (рисунок 4).

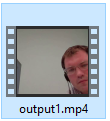


Рисунок 4 - Результат работы алгоритма при значениях 1.

1. Размер ядра 5, отклонение - 30, пороговое значение - 60 и площадью 50 (рисунок 5).

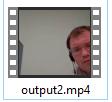


Рисунок 5 - Результат работы алгоритма при значениях 2.

**Листинг программ**

import cv2  
import numpy as np  
  
  
video = cv2.VideoCapture("ЛР4\_main\_video.mov",cv2.CAP\_ANY)  
  
def detect(kernel\_size, standard\_deviation, delta\_tresh, min\_area):  
  
 \_, frame = video.read()  
 img = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 img = cv2.GaussianBlur(  
 img, (kernel\_size, kernel\_size), standard\_deviation)  
  
 w = int(video.get(3))  
 h = int(video.get(4))  
  
 fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')  
 video\_writer = cv2.VideoWriter('output2.mp4', fourcc, 25, (w, h))  
  
 while True:  
 old\_img = img.copy()  
 flag, frame = video.read()  
 if not flag:  
 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, \_ = cv2.findContours(thresh,cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
 for contr in contors:  
 area = cv2.contourArea(contr)  
 if area >= min\_area:  
 video\_writer.write(frame)  
 break  
 video\_writer.release()  
  
  
kernel\_size = 5  
standard\_deviation = 30  
delta\_tresh = 60  
min\_area = 50  
detect(kernel\_size, standard\_deviation, delta\_tresh, min\_area)