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 выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д. Н. Баева

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

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

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

**Ход работы:**

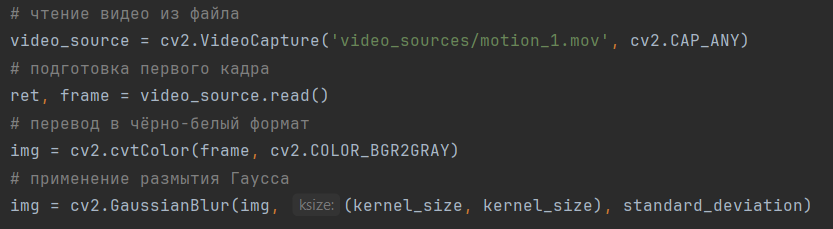
Задание 1 (самостоятельно). Реализовать метод, который читает видеофайл и записывает в один файл только ту часть видео, где в кадре было движение, можно воспользоваться примерами.

Задание 2 (самостоятельно). Провести эксперименты, выбирая различные значения параметров: размытие Гаусса, граница разделения для метода threshold, площадь минимального объекта, подобрать оптимальные значения параметров для данного видео.

В лабораторной работе необходимо было решить задачу детекции движения на видео, т. е. описать процесс обнаружения объектов в видеопотоке.

Решение задачи выполнялось строго в соответствие с описанием алгоритма в лабораторной работе.

В первую очередь, необходимо было начать чтения видеопотока из файла, прочесть первый кадр, перевести его в чёрно-белый цвет и применить размытие по Гауссу. Файл с видео располагается в соответствующей папке с видеоисточниками. Использовались стандартные функции, разобранные в предыдущих лабораторных работах. Фрагмент кода, выполняющий это, представлен на рисунке 1.

Рисунок 1 – Чтение и размытие изображения.

Далее необходимо было подготовить файл для записи – сперва были определены размеры кадра для записи при помощи стандартных функций get(cv2.CAP\_PROP\_FRAME\_WIDTH)) и get(cv2.CAP\_PROP\_FRAME\_HEIGHT)), а также формат видеофайла (был указан формат mp4). C помощью класса VideoWriter производится запись файла – результирующего видео.

Основная логика поиска детекции на видео заключена в работе цикла, завершающегося по завершении файла. Внутри этого цикла сперва выполняется копирование старого кадра. Это сохранение необходимо, чтобы использовать его в дальнейшем при вычислении разницы между кадрами. Следом выполняются стандартное чтение нового кадра, перевод его в чёрно-белый формат и применение размытия Гаусса. В случае, если кадр не прочтётся, то цикл завершает свою работу.

После этого функция cv2.absdiff(img, old\_img) вычисляет разницу между двумя кадрами. Вычисляется разница между текущим кадром img и предыдущим кадром old\_img. Разница между этими двумя кадрами показывает, какие пиксели были изменены между ними. Это поможет обнаружить движение или другие изменения на видео.

Следующим шагом необходимо найти пороговое (бинарное) изображение. Это изображение, в котором каждый пиксель принимает значение либо 0, либо 255, в зависимости от того, находится ли его интенсивность выше или ниже определенного порогового значения. Для его получения использовался метод, который уже применялся в лабораторных работах ранее для построения маски изображения ­- cv2.threshold(). Первым параметром является разница между текущим кадром и предыдущим кадром, вторым – пороговое значение, которое используется для классификации значений пикселей, третий параметр 255 - это максимальное значение, которое присваивается значениям пикселей, превышающим пороговое значение и последний параметр cv2.THRESH\_BINARY - это тип порогового значения, который указывает, что пороговое значение должно быть вычислено как бинарное значение. После получения бинарного изображения с помощью метода cv2.findContours() были найдены контуры объектов на изображении. Метод принимает входное изображение и несколько параметров, таких как режим извлечения контуров и метод аппроксимации контуров и возвращает список контуров, найденных на изображении, и иерархию контуров. Контур — это кривая (массив точек), которая представляет собой границу объекта на изображении. Он может быть использован для определения формы объекта на изображении или для обнаружения изменений между двумя кадрами видео. Соответствующие функции представлены на рисунке 2.

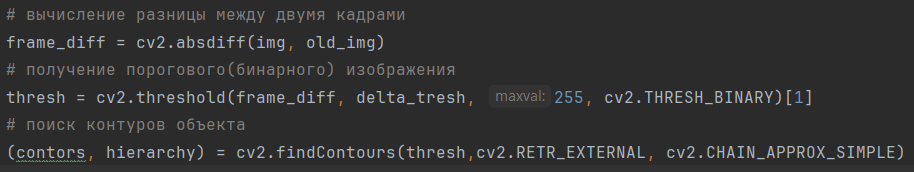


Рисунок 2 – Обработка кадров видео.

С помощью цикла for выполняется прохождение по полученному списку контуров. Вычисляется параметр для сравнения контуров (cv2.counturArea())- вычисляется площади контура. Затем площадь каждого контура сравнивается с наперед заданным параметром min\_area. Если площадь контура меньше, чем min\_area, то движения не было и цикл продолжается. Если площадь контура больше или равна min\_area, то было обнаружено движение и текущий кадр записывается в файл с помощью метода write() и обработка завершается(рисунок 3).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3 – Обработка полученных контуров.

На рисунках 4–8 отображены результаты экспериментов при различных параметрах размытия Гаусса, границ разделения для метода threshold, площади минимального объекта.

****

Рисунок 4 – Результат работы алгоритма при значении ядра 3, отклонением 50, пороговым значением 60 и площадью 20.



Рисунок 5 – Результат работы алгоритма при значении ядра 3, отклонением 50, пороговым значением 40 и площадью 5.



Рисунок 6 – Результат работы алгоритма при значении ядра 5, отклонением 50, пороговым значением 60 и площадью 20.



Рисунок 7 – Результат работы алгоритма при значении ядра 5, отклонением 30, пороговым значением 50 и площадью 15.



Рисунок 8 – Результат работы алгоритма при значении ядра 3, отклонением 50, пороговым значением 40 и площадью 5.

**Листинг программы**

Файл Motion\_Detection\_On\_Video.py

import cv2  
  
i = 0  
def main(kernel\_size, standard\_deviation,delta\_tresh, min\_area):  
 global i  
 i += 1  
  
 # чтение видео из файла  
 video\_source = cv2.VideoCapture('video\_sources/motion\_1.mov', cv2.CAP\_ANY)  
 # подготовка первого кадра  
 ret, frame = video\_source.read()  
 # перевод в чёрно-белый формат  
 img = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 # применение размытия Гаусса  
 img = cv2.GaussianBlur(img, (kernel\_size, kernel\_size), standard\_deviation)  
  
 # подготовка записи файла  
 # определение размера кадра при записи  
 w = int(video\_source.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  
 h = int(video\_source.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  
 # определение формата видеофайла  
 fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')  
 # запись видео в файл  
 video\_writer = cv2.VideoWriter( 'result\_videos/result\_' + str(i) + '.mp4', fourcc, 25, (w, h))  
  
 # цикл, завершающийся по завершению файла  
 while True:  
 print('Выполняется обработка видео...')  
 # копирование старого кадра  
 old\_img = img.copy()  
 # чтение нового кадра, перевод в чёрно-белый формат и фильтр Гаусса  
 is\_ok, frame = video\_source.read()  
 if not is\_ok:  
 break  
 img = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 img = cv2.GaussianBlur(img, (kernel\_size, kernel\_size), standard\_deviation)  
  
 # вычисление разницы между двумя кадрами  
 frame\_diff = cv2.absdiff(img, old\_img)  
 # получение порогового(бинарного) изображения  
 thresh = cv2.threshold(frame\_diff, delta\_tresh, 255, cv2.THRESH\_BINARY)[1]  
 # поиск контуров объекта  
 (contors, hierarchy) = cv2.findContours(thresh,cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
  
 # прохождение по списку контуров  
 for contr in contors:  
 # вычисление площади контура  
 area = cv2.contourArea(contr)  
 # если контур больше, чем наперед заданный параметр - было движение  
 if area < min\_area:  
 continue  
 # запись кадра в файл  
 video\_writer.write(frame)  
 video\_writer.release()  
 print('Обработка завершена.')  
  
kernel\_size = 3  
standard\_deviation = 50  
delta\_tresh = 60  
min\_area = 20  
main(kernel\_size, standard\_deviation,delta\_tresh,min\_area)  
  
# эксперименты с параметрами  
kernel\_size = 3  
standard\_deviation = 50  
delta\_tresh = 40  
min\_area = 5  
main(kernel\_size, standard\_deviation,delta\_tresh,min\_area)  
  
kernel\_size = 5  
standard\_deviation = 50  
delta\_tresh = 70  
min\_area = 30  
main(kernel\_size, standard\_deviation,delta\_tresh,min\_area)  
  
kernel\_size = 5  
standard\_deviation = 30  
delta\_tresh = 50  
min\_area = 15  
main(kernel\_size, standard\_deviation,delta\_tresh,min\_area)  
  
kernel\_size = 5  
standard\_deviation = 50  
delta\_tresh = 60  
min\_area = 20  
main(kernel\_size, standard\_deviation,delta\_tresh,min\_area)