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

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

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

Лабораторная работа №5

Выявление 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.

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

В начале алгоритма мы реализуем чтение из файла при помощи уже знакомых команд.Далее мы переводим наш кадр в чернобелый цвет так же, как и в прошлых лабах, и используем cv2.GaussianBlur для размытия Гаусса.

После чего идет подготовка файла для записи, используя команды

-**fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')**: Здесь устанавливается кодек для видеозаписи. Кодек - это программное обеспечение или алгоритм, которое используется для сжатия и декомпрессии видеоданных. В данном случае, **'mp4v'** представляет кодек MPEG-4 Part 2, который обеспечивает хорошее качество видео и относительно низкое использование места. Функция **cv2.VideoWriter\_fourcc** принимает строку с четырьмя символами (fourcc), представляющими кодек, и возвращает числовое значение, которое будет использоваться для кодека.

-**video\_writer = cv2.VideoWriter(r'.\LR5\_output ' + str(i) + '.mp4', fourcc, 144, (w, h))**: Здесь создается объект для записи видео с использованием указанного кодека. Параметры этой команды включают:

* Путь к выходному видеофайлу
* **fourcc**: Кодек, который был установлен ранее.
* **144**: Это количество кадров в секунду (fps), которое будет установлено для выходного видео.
* **(w, h)**: Это кортеж, который определяет размер кадра видео в пикселях.

Далее начинается цикл, который завершиться по завершению файла, внутри этого цикла мы копируем старый кадр, читаем новый кадр,переводим в чёрно-белый цвет и применяем размытие Гаусса при помощи уже описанных команд.

Следующий шаг - найти разницу между двумя кадрами в отдельный фрейм используя команду diff **= cv2.absdiff(img, old\_img)**. Она используется для вычисления абсолютной разницы между двумя изображениями **img** и **old\_img**.

Вот как это работает:.

1. **img**: Это текущий кадр, который был считан из видеопотока и обработан
2. **old\_img**: Это предыдущий кадр

**cv2.absdiff** выполняет поэлементное абсолютное вычитание между двумя изображениями, что означает, что для каждого пикселя в каждом канале (в данном случае, оттенков серого) вычисляется абсолютная разница между соответствующими пикселями в **img** и **old\_img**.

Результат этой операции **diff** будет представлять собой новое изображение, в котором каждый пиксель будет содержать абсолютное значение разницы между пикселями в **img** и **old\_img**. Это изображение можно использовать для выделения областей изменений между двумя последовательными кадрами видеопотока, что часто полезно в приложениях обнаружения движения или выделения объектов.

Далее проводим операцию двоичного разделения для фрейма используя cv2.threshold. Эта команда используется для применения бинаризации к изображению diff с целью выделения областей, в которых разница между текущим и предыдущим кадрами превышает пороговое значение delta\_thresh.

Параметры:

1. **diff**: Это изображение, которое представляет собой разницу между текущим и предыдущим кадрами видеопотока.
2. **delta\_thresh**: Это пороговое значение. Все пиксели в **diff**, чьи значения абсолютной разницы превышают **delta\_thresh**, будут преобразованы в белый цвет (255), а все остальные пиксели (те, чьи значения абсолютной разницы меньше или равны **delta\_thresh**) будут преобразованы в черный цвет (0) после бинаризации.Более высокое значение delta\_thresh будет более консервативным и выделит только очень яркие изменения. Если установить **delta\_thresh** на низкое значение, то больше пикселей будут считаться измененными, даже если разница между текущим и предыдущим кадрами небольшая. Это может привести к более широкому выделению измененных областей.Маленькие артефакты или изменения в освещении могут быть интерпретированы как изменения, что может привести к ложным срабатываниям.
3. **cv2.THRESH\_BINARY**: Это флаг, указывающий тип бинаризации, который будет применен к **diff**. В данном случае, **cv2.THRESH\_BINARY** означает, что пиксели, превышающие порог **delta\_thresh**, будут установлены в максимальное значение (255), а пиксели, не превышающие порог, будут установлены в минимальное значение (0).

Результат выполнения этой команды **thresh** представляет собой бинарное изображение, где белые области обозначают места, где разница между кадрами была больше **delta\_thresh**, а черные области обозначают места, где разница была меньше или равна **delta\_thresh**.

После чего необходимо найти контуры объектов для фрейма используя cv2.findContours. Эта команда используется для нахождения контуров (границ объектов) в бинарном изображении **thresh**.

Параметры:

1. **thresh**: Это бинарное изображение, в котором пиксели имеют только два значения: 0 (черный) или 255 (белый).
2. **cv2.findContours**: Это функция из библиотеки OpenCV, которая выполняет поиск контуров в бинарных изображениях. Она принимает следующие аргументы:
   * **thresh**: Бинарное изображение, в котором нужно найти контуры.
   * **cv2.RETR\_EXTERNAL**: Этот флаг определяет, какие контуры следует извлекать. **cv2.RETR\_EXTERNAL** означает, что нужно извлекать только внешние контуры, то есть контуры, которые окружают объекты, без внутренних контуров объектов.
   * **cv2.CHAIN\_APPROX\_SIMPLE**: Этот флаг определяет способ хранения контуров. **cv2.CHAIN\_APPROX\_SIMPLE** означает, что контуры будут храниться в более компактной форме, описывая только точки вершин контуров, а не все пиксели на контуре. Это экономит память и упрощает анализ контуров, если не нужна подробная информация о форме контуров.

Результат выполнения **cv2.findContours** возвращает два значения:

* **contours**: Это список контуров, найденных на изображении. Каждый контур представлен как список точек, которые образуют контур.
* **hierarchy**: Это информация о иерархии контуров, если она доступна. В случае использования **cv2.RETR\_EXTERNAL**, иерархия обычно не используется, и значение **hierarchy** может быть проигнорировано.

После выполнения этой команды, получим список **contours**, который содержит все контуры на бинарном изображении **thresh**.

Следующий шаг - пройтись по контурам объектов для фрейма и найти контур площадью большей, чем заданный параметр-cv2.contourArea: если такой контур найден, значит было движение-записать кадр в файл. Команда **area = cv2.contourArea(contour)** используется для вычисления площади контура, который представлен в виде списка точек в библиотеке OpenCV. Результат выполнения **cv2.contourArea** будет числовым значением, представляющим площадь контура в квадратных пикселях. Площадь вычисляется путем подсчета пикселей, которые находятся внутри контура. Параметр **min\_area** используется для определения минимальной площади контура, при которой кадр будет записан в выходное видео. Увеличение **min\_area** будет означать, что только более крупные объекты или изменения будут записаны, что может быть полезно для фильтрации маленьких шумовых изменений. Уменьшение **min\_area** позволит сохранить более мелкие объекты и детали, но также может включить в себя шум или нежелательные мелкие детали.

[Изображение выглядит как текст, программное обеспечение, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание](LR5_output%202.mp4)

Рисунок 1 – получившееся видео(работает при нажатии).

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

Для 1 эксперимента были подобраны параметры **kernel\_size = 3, standard\_deviation = 80, delta\_thresh = 60, min\_area = 20**

Результаты: Размытие средней силы, умеренный порог для детекции изменений и минимальная площадь объекта для сохранения. Кадры кажутся замедленными из-за размытия и умеренного порога.

Вывод: Этот набор параметров может быть полезным для выделения средних по размеру изменений и снижения шума, но он может уменьшить скорость видео.

[Изображение выглядит как текст, программное обеспечение, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание](LR5_output%201.mp4)

Рисунок 2 – видео с медленными кадрами(работает при нажатии).

Для второго эксперимента были подобраны параметры **kernel\_size = 11**, **standard\_deviation = 60**, **delta\_thresh = 60**, **min\_area = 20.**

Результаты: Увеличенное ядро размытия и уменьшенное стандартное отклонение. Кадры могут оставаться плавными из-за размытия, но детекция изменений остается приемлемой.

Вывод: Этот набор параметров может предоставить хороший баланс между сглаживанием и точностью обнаружения изменений.Этот набор показал лучший результат.

[Изображение выглядит как текст, программное обеспечение, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание](LR5_output%202.mp4)

Рисунок 3 – лучший результат(работает при нажатии).

Для 3 эксперимента были выбраны параметры **kernel\_size = 9**, **standard\_deviation = 30**, **delta\_thresh = 2**, **min\_area = 20**

Результаты: слишком маленькое значение **delta\_thresh**, что приводит к включению кадров с минимальными изменениями, которые, возможно, не нужны.

Вывод: Этот набор параметров слишком чувствителен к изменениям и может захватывать нежелательные детали.

[Изображение выглядит как текст, программное обеспечение, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание](LR5_output%203.mp4)

Рисунок 4 – видео с ненужными кадрами(работает при нажатии).

Для 4 эксперимента были подобраны параметры **kernel\_size = 3**, **standard\_deviation = 50**, **delta\_thresh = 60**, **min\_area = 1000**

Результаты: Очень большое значение **min\_area**, что приводит к тому, что сохраняются не все кадры с изменениями.

Вывод: Этот набор параметров установил слишком высокий порог для сохранения кадров, и как результат, множество изменений не были зафиксированы.

[Изображение выглядит как текст, программное обеспечение, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание](LR5_output%204.mp4)

Рисунок 5 – видео, в котором зафиксированы не все изменения

(работает при нажатии).

**Листинг программы**

**python**

import cv2  
  
i = 0  
  
  
# Основная функция для обработки видео  
def main(kernel\_size, standard\_deviation, delta\_thresh, min\_area):  
 global i  
 i += 1  
  
 # Открываем видеофайл для чтения  
 video = cv2.VideoCapture(r'ЛР5\_main\_video.mov', cv2.CAP\_ANY)  
  
 # Читаем первый кадр и преобразуем его в оттенки серого  
 #ret-Это булевая переменная, которая указывает, удалось ли успешно прочитать кадр из видеопотока.  
 #frame: Это переменная, которая содержит сам кадр как изображение.  
 ret, frame = video.read()  
 img = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
  
 # Применяем гауссово размытие для сглаживания шума  
 img = cv2.GaussianBlur(img, (kernel\_size, kernel\_size), standard\_deviation)  
  
 # Получаем ширину и высоту видеофайла в пикселях из видеопотока  
 # используем метод get объекта видеозахвата video, чтобы получить значение конкретного свойства видеопотока  
 w = int(video.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  
 h = int(video.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  
  
 # Устанавливаем кодек и создаем объект для записи видео  
 fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')  
 video\_writer = cv2.VideoWriter(r'.\LR5\_output ' + str(i) + '.mp4', fourcc, 144, (w, h))  
  
 while True:  
 # Сохраняем старый кадр, чтобы вычислить разницу между кадрами  
 old\_img = img.copy()  
 ok, frame = video.read()  
 if not ok:  
 break  
  
 # Преобразуем текущий кадр в оттенки серого и применяем гауссово размытие  
 img = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 img = cv2.GaussianBlur(img, (kernel\_size, kernel\_size), standard\_deviation)  
  
 # Вычисляем разницу между текущим и предыдущим кадрами  
 diff = cv2.absdiff(img, old\_img)  
  
 # Применяем бинаризацию, чтобы выделить измененные области  
 thresh = cv2.threshold(diff, delta\_thresh, 255, cv2.THRESH\_BINARY)[1]  
  
 # Находим контуры в бинарном изображении  
 (contours, hierarchy) = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
  
 # Если на кадре есть хотя бы один контур, площадь которого больше min\_area, записываем кадр  
 for contour in contours:  
 area = cv2.contourArea(contour)  
 if area < min\_area:  
 continue  
 video\_writer.write(frame)  
  
 # Закрываем объект записи видео  
 video\_writer.release()  
  
  
# Первый набор параметров  
# умеренное размытие и порог детекции для изменений, а также минимальную площадь объекта для сохранения.  
# замедленные кадры  
kernel\_size = 3  
standard\_deviation = 80  
delta\_thresh = 60  
min\_area = 20  
main(kernel\_size, standard\_deviation, delta\_thresh, min\_area)  
  
# Второй набор параметров  
#увеличение размера ядра и уменьшение стандартного отклонения-лучший вариант  
kernel\_size = 11  
standard\_deviation = 60  
delta\_thresh = 60  
min\_area = 20  
main(kernel\_size, standard\_deviation, delta\_thresh, min\_area)  
  
# Третий набор параметров  
#слишком маленькаое значение дельты-попали кадры с минимальными изменениями, которые нам не нужны  
kernel\_size = 9  
standard\_deviation = 30  
delta\_thresh = 2  
min\_area = 20  
main(kernel\_size, standard\_deviation, delta\_thresh, min\_area)  
  
# Четвертый набор параметров  
#очень большое значение min\_area-зафиксированы не все кадры изменений  
kernel\_size = 3  
standard\_deviation = 50  
delta\_thresh = 60  
min\_area = 1000  
main(kernel\_size, standard\_deviation, delta\_thresh, min\_area)