Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Кафедра информационных технологий и систем

**Программа обработки видео**

Лабораторная работа по дисциплине:

«Распознавание образов и обработка изображений»

Выполнил студент группы 8091:

\_\_\_\_\_\_\_\_\_\_ / Кузин И.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Принял преподаватель:

\_\_\_\_\_\_\_\_\_\_ / Назаров A. Г.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**Великий Новгород**

**2022**

1. **Цель работы**

Изучение основ обработки потока изображений и алгоритмов детекции движения.

На основе полученных знаний, разработать рабочую программу обработки видео.

1. **Описание среды окружения**

Для решения поставленной задачи использовалось следующее окружение:

* ОС: Windows 10
* Среда программирования: Qt версии 6.2.2
* Библиотека OpenCV 4.5.5
* Компилятор: Qt 6.2.3 MinGW 64-bit

1. **Описание работы программы-детектора движения**

С помощью программы-детектора движения возможно фиксировать движение любого объекта, в том числе, незаконное проникновение в квартиру, к примеру, камеру можно вставить в электронный дверной глазок, или фиксировать подозрительную активность вокруг личного автомобиля, организовать видеонаблюдение на даче и тому подобное.

Существует множество различных методов для реализации программы-детектора движения, при разработке программы для выполнения данной лабораторной работы был выбран метод фоновой сегментации с использованием возможностей библиотеки OpenCV.

Для данной реализации были задействованы следующие методы, согласно официальной документации OpenCV:

1. *createBackgroundSubtractorMOG2 (history, varTrashold, detectShadows) -* это алгоритм сегментации переднего плана/фона на основе смешанной гауссовой модели.

Параметры данного метода:

* *history* – количество кадров при выборки фонового изображения;
* *varTrashold* – пороговое значение для квадрата расстояния между пикселем и образцом, чтобы определить, близок ли пиксель к этому образцу. Этот параметр не влияет на фоновое обновление;
* *detectShadows* – если принимаемое значение **true**, алгоритм обнаружит тени и пометит их. Это снижает скорость работы алгоритма детекции, поэтому, если в реализации не нужна эта функция, необходимо установить для параметра значение **false**.

1. Переданное видео состоит из кадров, где для обработки каждого кадра применяется:

* *segmentor->apply(frame, fgmask)* –в результате выполнения данной функции, производится генерирование изображения в градациях серого с заполненным черным фоном, а часть переднего плана заполняется не блочными пикселями.Это достигается путем применения вычитателя сегментов с захваченным кадром и маской переднего плана, где мы получаем информацию о сцене и извлекаем передний план и фон.

1. Далее для того, чтобы избавиться от лишних шумов на полученной маске переднего плана и выделить интересующие нас объекты, последовательно применяются следующие методы:

* *cv::threshold(fgmask, fgmask, 25, 255, cv::THRESH\_BINARY) -* производитя удаление лишних темных шумов с маски переднего плана;
* *cv::Mat kernel = cv::getStructuringElement(cv::MORPH\_RECT, cv::Size(noise\_size, noise\_size)) -* метод в результате своей работы возвращает структурирующий элемент определенного размера и формы для дальнейших операций;
* *cv::erode(fgmask, fgmask, kernel) -* Выполняем размытие с помощью структурирующего элемента;
* *cv::dilate(fgmask, fgmask, kernel, cv::Point(-1,-1), 3) –* метод, на входе принимает изображение и расширяет его на выходе, используя указанный структурирующий элемент, который определяет форму окрестности пикселя, по которой берется максимум.

1. После обработки кадра, необходимо определить контуры объектов на маске используются данные методы:

* *cv::findContours(fgmask, contours, cv::RETR\_TREE*, *cv::CHAIN\_APPROX\_SIMPLE) –* поиск контуров использует для работы монохромное изображение, так, что все пиксели переданного изображения с ненулевым цветом будут интерпретироваться как 1, а все нулевые останутся нулями.В случае обнаружения контуров, возможно выполнить предположение, что движение было успешно обнаружено.

1. В заключительном этапе работы алгоритма, для каждого контура, на карте захвата происходит отрисовка прямоугольников, отмечающих обнаруженное движение.
2. **Скриншоты интерфейса и работы программы.**

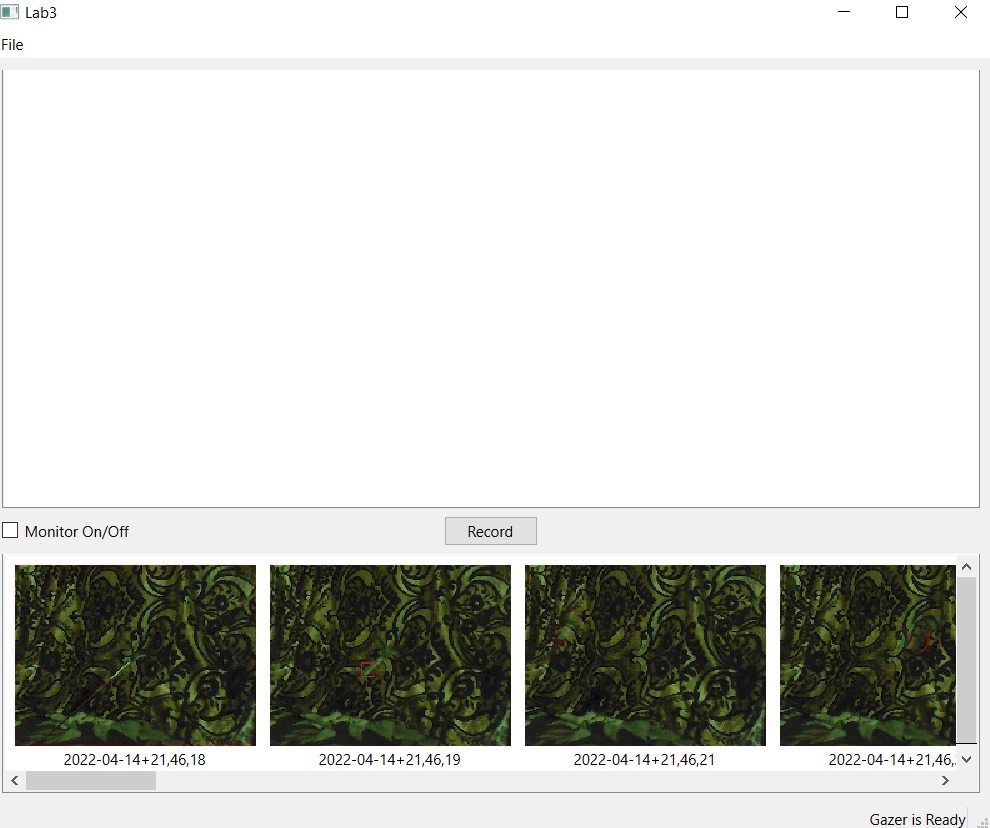


Рисунок 1. Интерфейс главного экрана программы

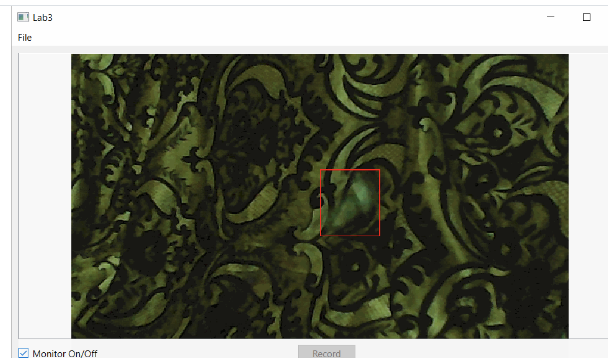


Рисунок 2. Детекция обнаруженного движения на загруженном видеофайле

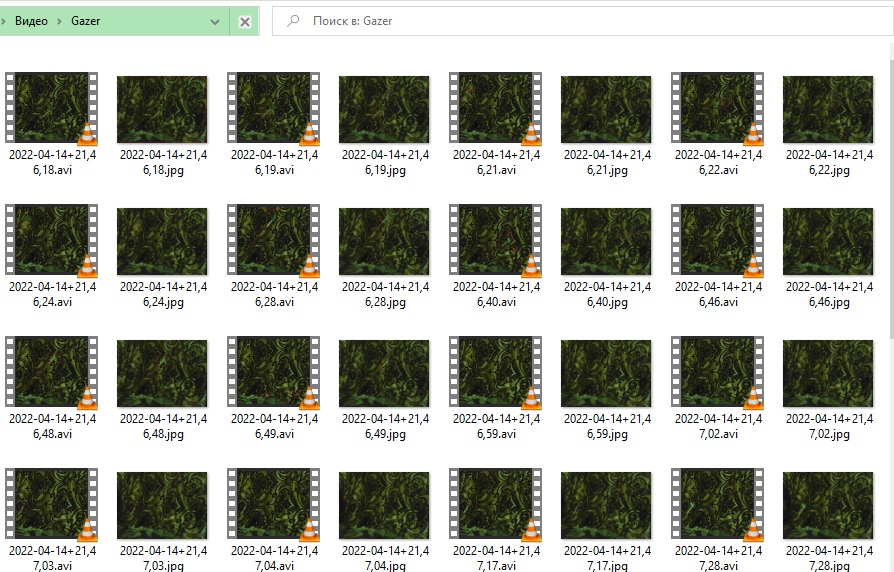


Рисунок 3. Сохраненные видеофайлы в процессе работы программы

1. **Производимые математические расчеты**

* Наиболее распространенный способ усреднить набор данных — взять его среднее значение. Для вычисления среднего значения пикселя (mean) производятся следующие математические расчеты:

В данной формуле:

* – это используемое значение пикселя для одного канала;
* n – это общее количество пикселей в изображении.
* Среднеквадратичное отклонение - наиболее распространённый показатель рассеивания значений случайной величины относительно её математического ожидания (аналога среднего арифметического с бесконечным числом исходов). Обычно он означает квадратный корень из дисперсии случайной величины, но иногда может означать тот или иной вариант оценки этого значения. Для вычисления среднеквадратичного отклонения значений пикселя (standard deviation) производятся следующие математические расчеты:
* Описание разницы между понятиями точность и кучность:

Под точностью понимают степень соответствия результатов, полученных в процессе исследований, измерений, экспериментов, их истинным значениям.

Кучность — это показатель соответствия результатов, полученных в процессе исследований, измерений, экспериментов, относительно друг друга.

Разницей между данными терминами будет являться то, что точность является показателем того, насколько матожидание близко к истинному, а кучность то, какие результаты имеет разброс (чем выше разброс, тем выше среднеквадратичное отклонение).

1. **Ответы на вопросы**
2. Возможность обнаруживать движение из готового видеофайла, вместо камеры есть. Для этого необходимо было бы загрузить в программу готовый видеофайл, который будет покадрово считываться и сам процесс обнаружения образа будет аналогичным, что и в трансляции, в режиме реального времени с камеры.
3. Возможность выполнять работу по обнаружению движения в потоке, отличном от потока захвата видео есть, но это значительно усложнит работу алгоритмов программы.

Для этого потребуется создание связи между двумя потоками, т.к. первому потоку, который производит захват изображения, необходимо будет ожидать окончание обработки предыдущего кадра вторым потоком, чтобы в дальнейшем передать на обработку следующий кадр.

В результате, пользователю все равно потребуется выдавать изображение с распознанным образом, поэтому в разделении алгоритмов программы на потоки нет необходимости, т.к. данный механизм приведет только к усложнению реализации разработки программы.

1. **Исходники созданной программы**

Исходный код данного проекта доступен к просмотру на GitHub по ссылке:

https://github.com/ivan-kuzin/Pattern-recognition/tree/main/Lab\_3 .

Для запуска программы, необходимо запустить файл ***Lab3.exe*** из каталога release. (Дополнительно, каталог приложен в формате архива).

1. **Вывод**

В ходе выполнения данной лабораторной работы, мною были изучены основы обработки потока изображений и алгоритмы детекции движения, а также реализована программа с использованием полученных знаний.