Лабораторная работа №6:
Обнаружение движения

Цель:

Целью данной работы является изучение методик обнаружения движения в видеопотоке.

План работ:

Необходимо разработать приложение Windows Forms, способное осуществлять:

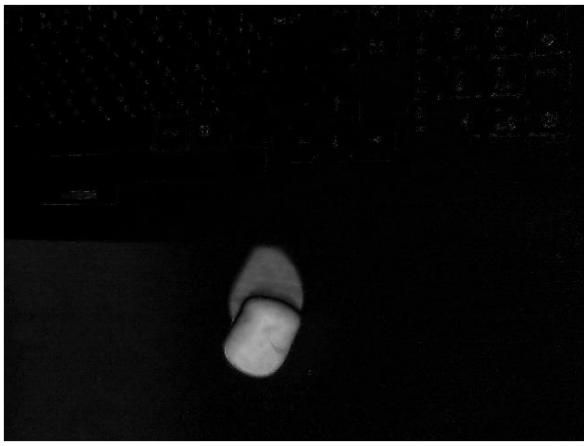
1. обнаружение подвижных объектов в видеопотоке.

Обнаружение движения:

В сущности, перемещение объекта на последовательности видеокадров можно описать как изменение цвета группы пикселей изображения относительно предыдущего или эталонного. Самым простым способом нахождения этого изменения является вычитание изображений по модулю. В местах, оставшихся неизменными, результат будет равен нулю, в местах, где произошли изменения, результат будет отличен от нуля.

Как правило, разность по модулю определяется для изображений в градациях серого. Ниже приведен пример текущего изображения (слева) и фона (справа):

Результат вычитания изображений по модулю:



Вычитание по модулю производится следующим образом:

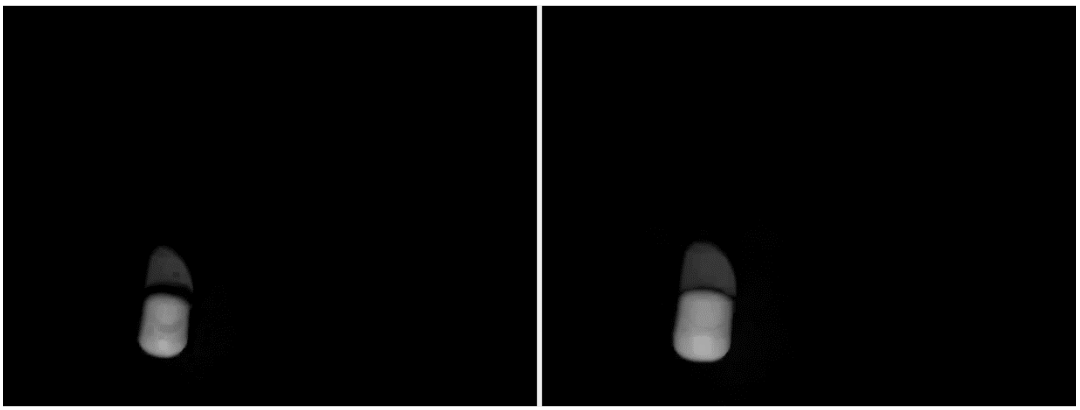
```
var diff = background.AbsDiff(frame);
```

Где `frame` – один из кадров видеопотока, а `background` – фон.

Однако, как можно заметить на изображении, помимо объекта, недавно появившегося в кадре, после вычитания остались мелкие шумы от бликов, погрешностей работы видеокамеры и т.д. Избавиться от них можно путем последовательного сужения и расширения светлых областей (при этом на изображении остаются только светлые области значительного размера):

```
diff.Erode(3);
diff.Dilate(4);
```

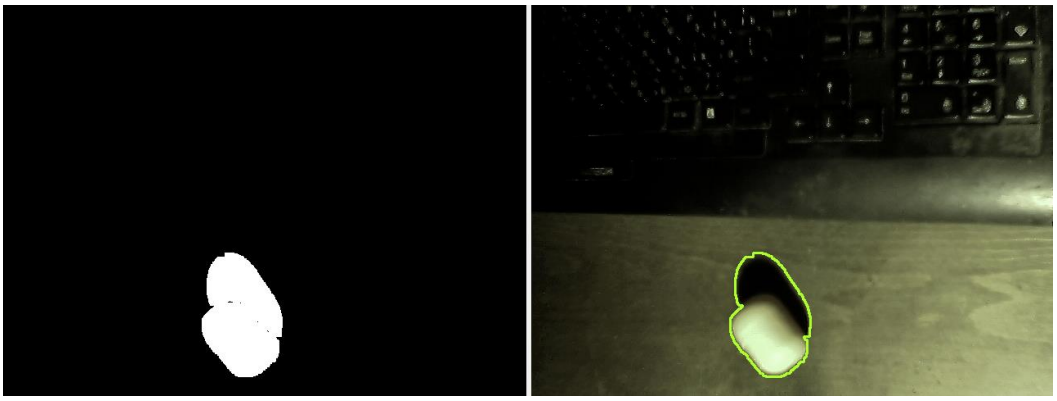
Результат сужения светлых областей (слева) и последующего их расширения (справа):



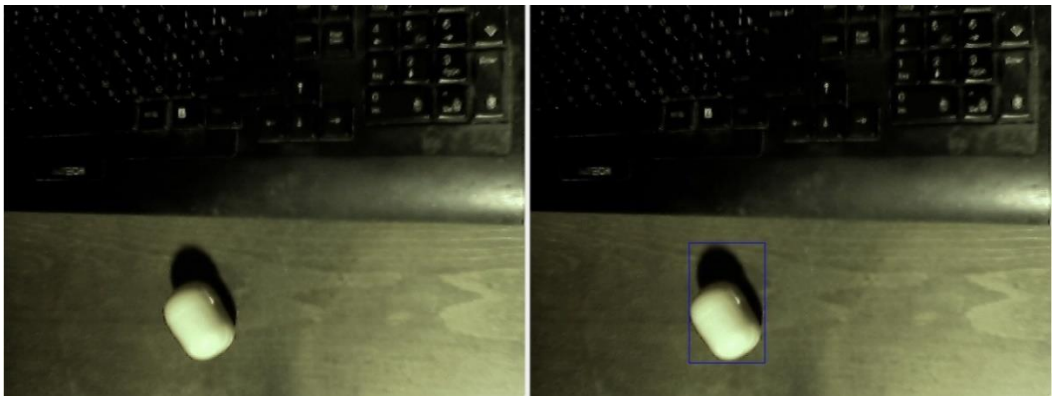
Альтернативным вариантом устранения мелких шумов может быть применение размытия:

```
var result = diff.SmoothMedian(5);
```

После обнаружения искомой области можно провести пороговое преобразование (слева) и найти её контур (справа):

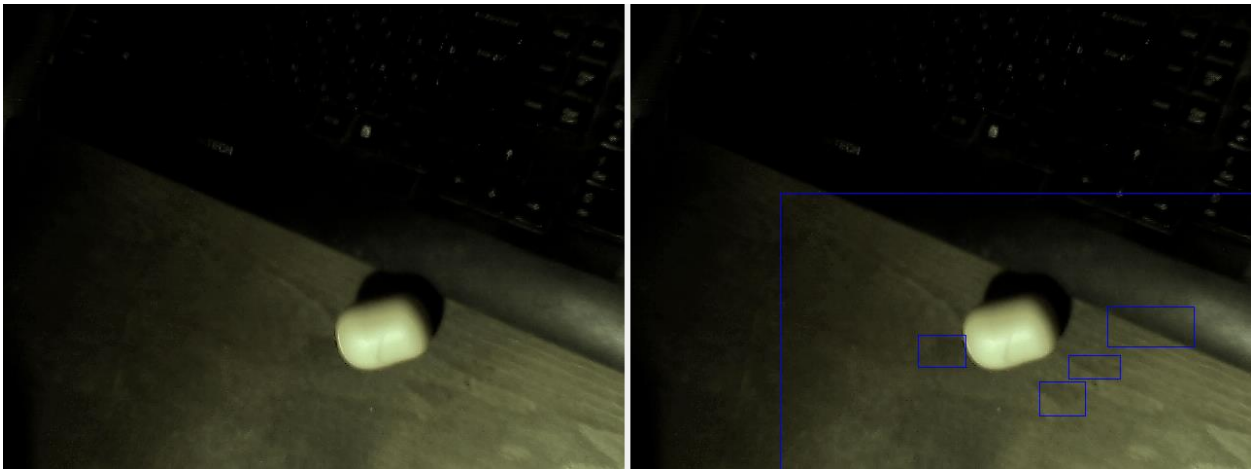


В завершение, можно обозначить найденную область, нарисовав её ограничивающий объём на исходном изображении:



Однако, подобная методика подходит только для сцен со статичным окружением. В случае изменения освещения, появлении бликов и отражений, изменении позиции камеры, возможно возникновение ложнопозитивных срабатываний.

Пример ложнопозитивных срабатываний при изменении освещённости сцены:



Библиотека OpenCV содержит реализацию некоторых алгоритмов вычисления актуального фонового изображения с учётом изменения освещённости, позиционирования теней, появления отражений и тому подобных явлений. В рамках лабораторной работы предлагается использовать улучшенную адаптивную модель смешивания (improved adaptive Gaussian mixture model for background subtraction).

Класс, реализующий данный алгоритм, называется `BackgroundSubtractorMOG2`. Объект этого класса можно объявить и инициализировать следующим образом:

```
BackgroundSubtractorMOG2 subtractor = new BackgroundSubtractorMOG2(1000, 32, true);
```

Объект класса `BackgroundSubtractorMOG2` может содержать следующие параметры:

history – длительность истории, по умолчанию 500;

varTreshold – количество компонент, участвующих в смешивании, по умолчанию 16;

shadowDetection – если установлено в true, алгоритм обнаруживает и отмечает тени, по умолчанию true.

Получение маски изменений можно записать как:

```
var foregroundMask = grayFrame.CopyBlank();
subtractor.Apply(grayFrame, foregroundMask);
```

где:

`grayFrame` – кадр видеопотока в градациях серого; передаётся для обновления информации о фоне;

`foregroundMask` – получаемая маска изменений.

Пример видеокадра движения автомобилей и маски изменений:

Как видно из примера, маска изменений получилась зашумленной и «рваной»: светлые области, соответствующие движущимся автомобилям, не имеют четких границ и содержат внутри себя более мелкие части (стекла, номерные знаки и др.). Для корректного определения движущихся объектов достаточно удаления шумов одним из ранее рассмотренных способов, но для получения более

совершенного результата предлагается использовать фильтр на основе морфологических операции: открытие, закрытие, расширение, сжатие и др. В качестве основы для фильтра можно использовать следующий пример кода:

Порядок использования и входные параметры морфологических операций могут различаться. Более подробное описание морфологических операций можно найти по ссылке: https://docs.opencv.org/3.1.0/d9/d61/tutorial_py_morphological_ops.html

```
private Image<Gray, byte> FilterMask(Image<Gray, byte> mask)
{
    var anchor = new Point(-1, -1);
    var borderValue = new MCvScalar(1);

    // создание структурного элемента заданного размера и формы для морфологических операций
    var kernel = CvInvoke.GetStructuringElement(ElementShape.Ellipse, new Size(3, 3), anchor);

    // заполнение небольших тёмных областей
    var closing = mask.MorphologyEx(MorphOp.Close, kernel, anchor, 1, BorderType.Default,
                                                                        borderValue);

    // удаление шумов
    var opening = closing.MorphologyEx(MorphOp.Open, kernel, anchor, 1, BorderType.Default,
                                                                      borderValue);

    // расширение для слияния небольших смежных областей
    var dilation = opening.Dilate(7);

    // пороговое преобразование для удаления теней
    var threshold = dilation.ThresholdBinary(new Gray(240), new Gray(255));

    return threshold;
}
```

Результат фильтрации:

Как видно из примера, светлые области, соответствующие движущимся автомобилям, стали более однородными.

Следующим шагом является нахождение контуров. В данном случае для нахождения контуров рекомендуется использовать следующие параметры:

```
CvInvoke.FindContours(
    foregroundMask,
    contours,
    null,
    RetrType.External, // получение только внешних контуров
    ChainApproxMethod.ChainApproxTc89L1); // применение одной из разновидностей
                                          // алгоритма аппроксимации цепочки Teh-Chin
```

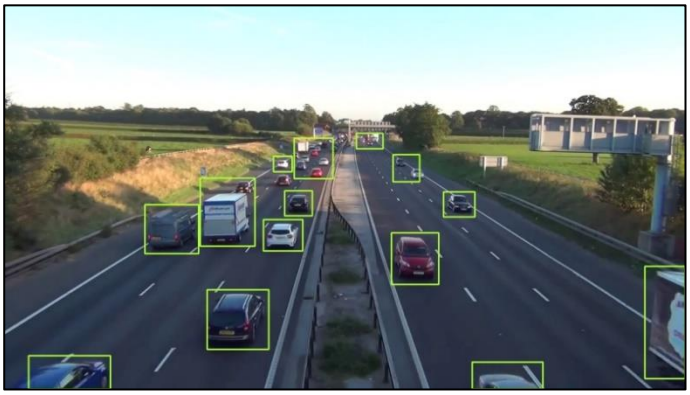
Для отсечения незначительных изменений можно добавить ограничение минимальной площади контуров:

```
if (CvInvoke.ContourArea(contours[i]) > 700)
```

Завершающим этапом будет нахождение ограничивающих объемов и отрисовка:

```
Rectangle boundingRect = CvInvoke.BoundingRectangle(contours[i]);
resultImage.Draw(boundingRect, new Bgr(Color.GreenYellow), 2);
```

Пример работы `BackgroundSubtractorMOG2` на видео «Road traffic video for object recognition»:

Захват видеопотока из файла:

Захват видеопотока из файла может быть выполнен аналогично захвату видеопотока с веб-камеры, с учетом того, что при создании объекта класса VideoCapture необходимо передать строку с именем файла:

```
private VideoCapture capture;
...
capture = new VideoCapture(fileName);
```

После чего можно получить данные о видео, например, частоту кадров:

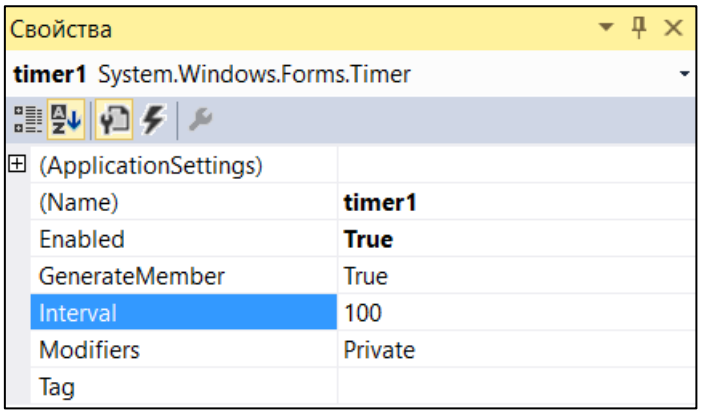
```
var frameRate = capture.GetCaptureProperty(CapProp.Fps);
```

Получить видеокадр можно следующим образом:

```
Mat frame = _capture.QueryFrame();
```

frame – может возвращать null, если кончились кадры, либо видео не было загружено.

Чтобы добиться частоты воспроизведения кадров, близкой к реальной, рекомендуется выводить кадры по таймеру. Стандартным компонентом Windows.Forms является компонент Timer:



Свойство Interval задает время между отсчетами таймера. В данном случае, время между отсчетами является обратной величиной относительно частоты кадров.

Задание:

Реализовать программное средство, позволяющее отображать в одном окне два изображения: «оригинальное» слева и «результат обработки» справа. Реализовать интерфейс, позволяющий по нажатию на соответствующие кнопки выполнять следующие операции:

1. обнаружение и обозначение перемещающихся объектов в видеопотоке с веб-камеры;
2. обнаружение и обозначение перемещающихся объектов в видеопотоке из видеофайла.