Лабораторная работа №1:
Использование библиотеки OpenCV в среде разработки Visual Studio

Цель:

Целью данной работы является проверка остаточных навыков программирования на языке высокого уровня C#, а также навыков использования среды разработки Microsoft Visual Studio.

Задание:

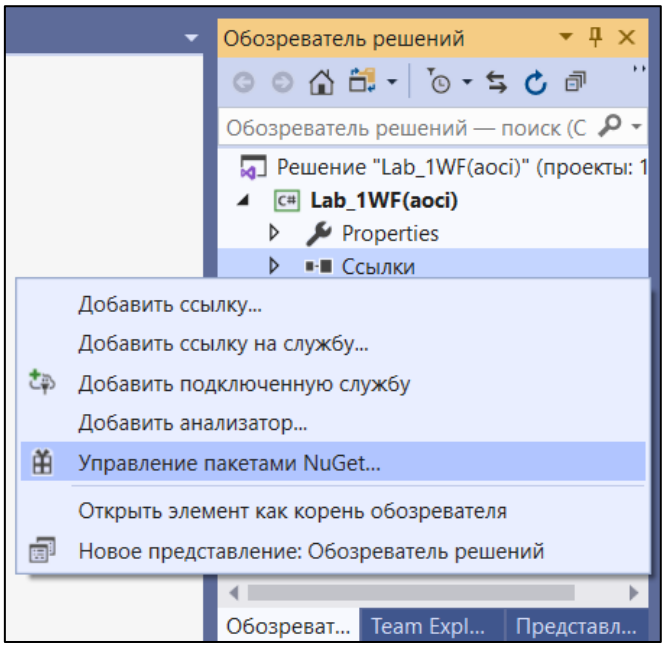
Необходимо разработать приложение WindowsForms, способное:

1. осуществлять выбор и загрузку изображения;
2. отображать изображение на экране;
3. обрабатывать изображение при помощи эффекта Cell Shading.

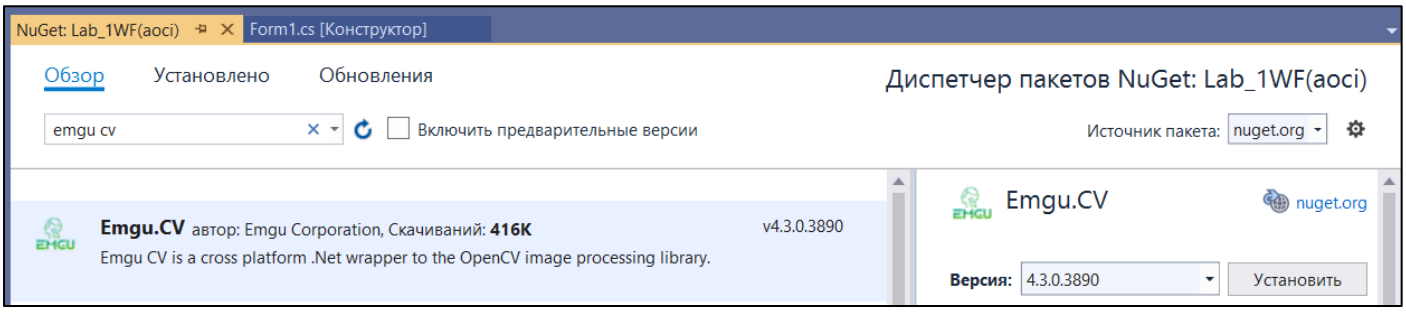
Справочная информация:
Среда разработки Microsoft Visual Studio 2019 Community доступна для скачивания на официальном сайте корпорации Microsoft по адресу: http://www.visualstudio.com/downloads/download-visual-studio-vs#DownloadFamilies_4
Библиотека OpenCV доступна по адресу: https://opencv.org/releases.html
Библиотека Emgu.CV, позволяющая использовать OpenCV в среде .NET, доступна по адресу: http://www.emgu.com/wiki/index.php/Main_Page

Создание приложения:

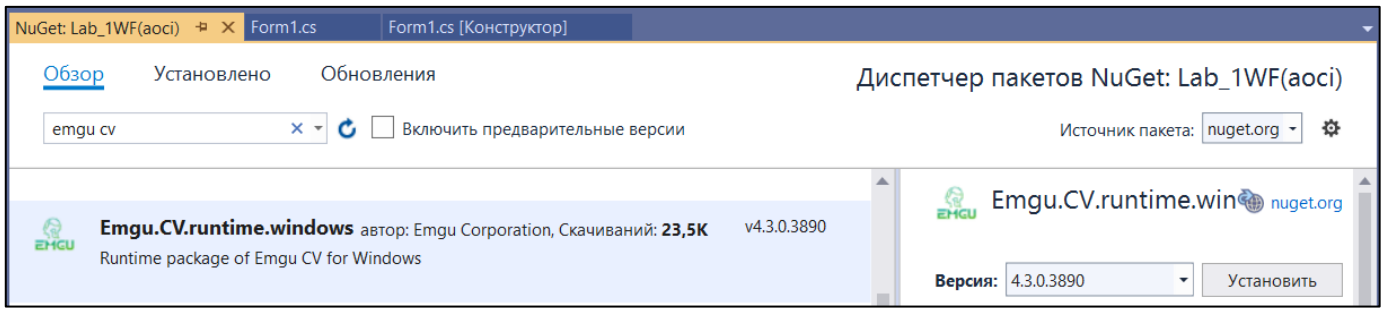
Создайте проект «Приложение WindowsForms (.NET Framework)». В контекстном меню проекта выберите «Управление пакетами NuGet».



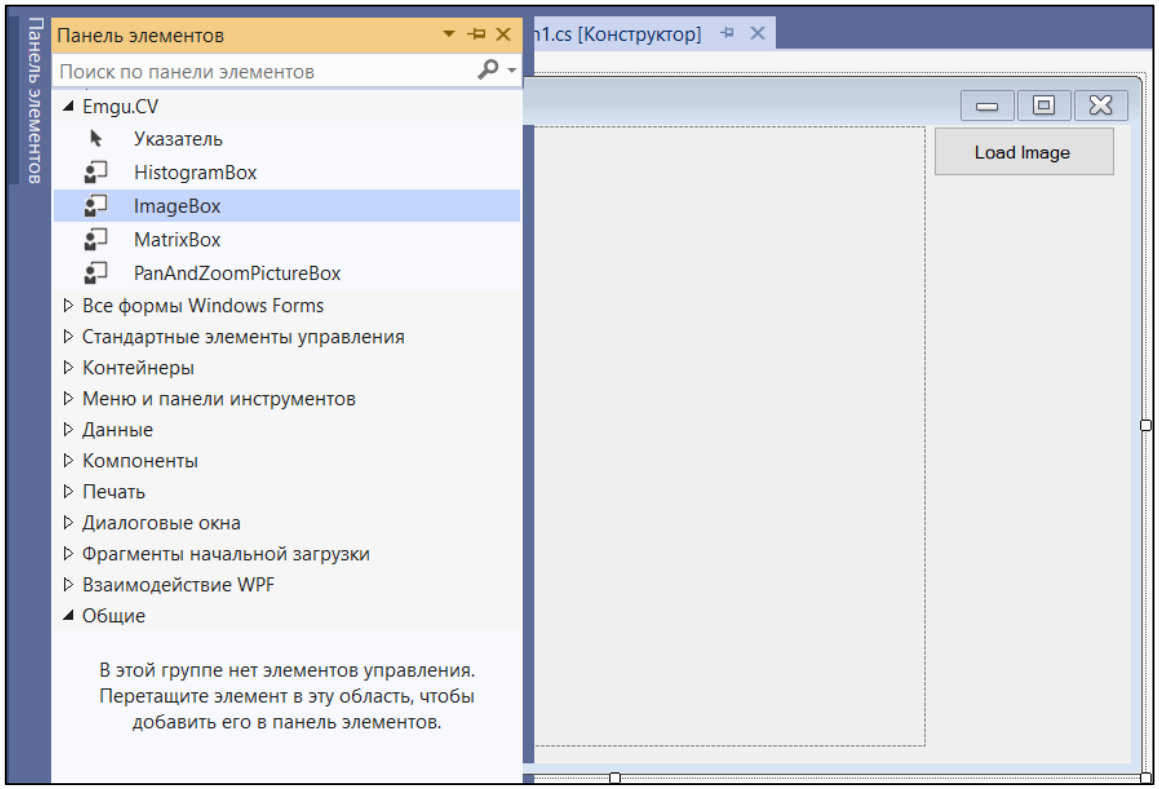
В диспетчере пакетов NuGet найдите и установите Emgu.CV:



В дополнение, установите среду исполнения Emgu.CV.runtime.windows:

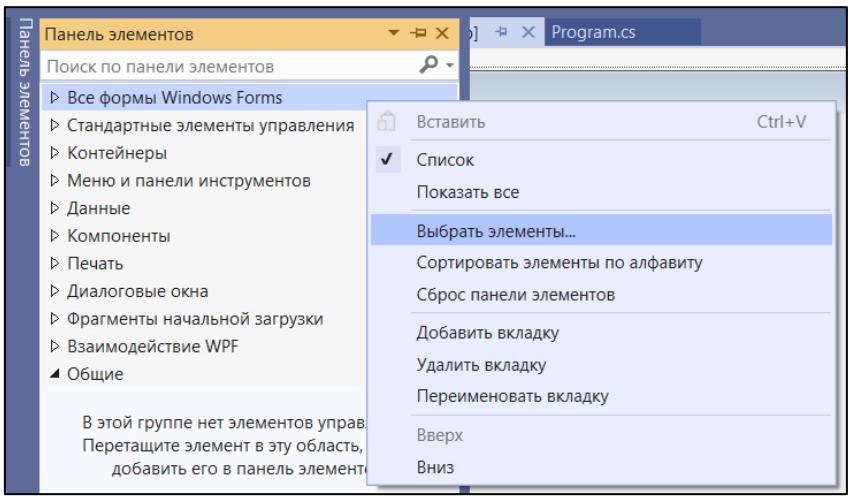


Для отображения изображений на форме используется компонент ImageBox. Найдите этот компонент в панели элементов, перетащите на форму и задайте необходимый размер:

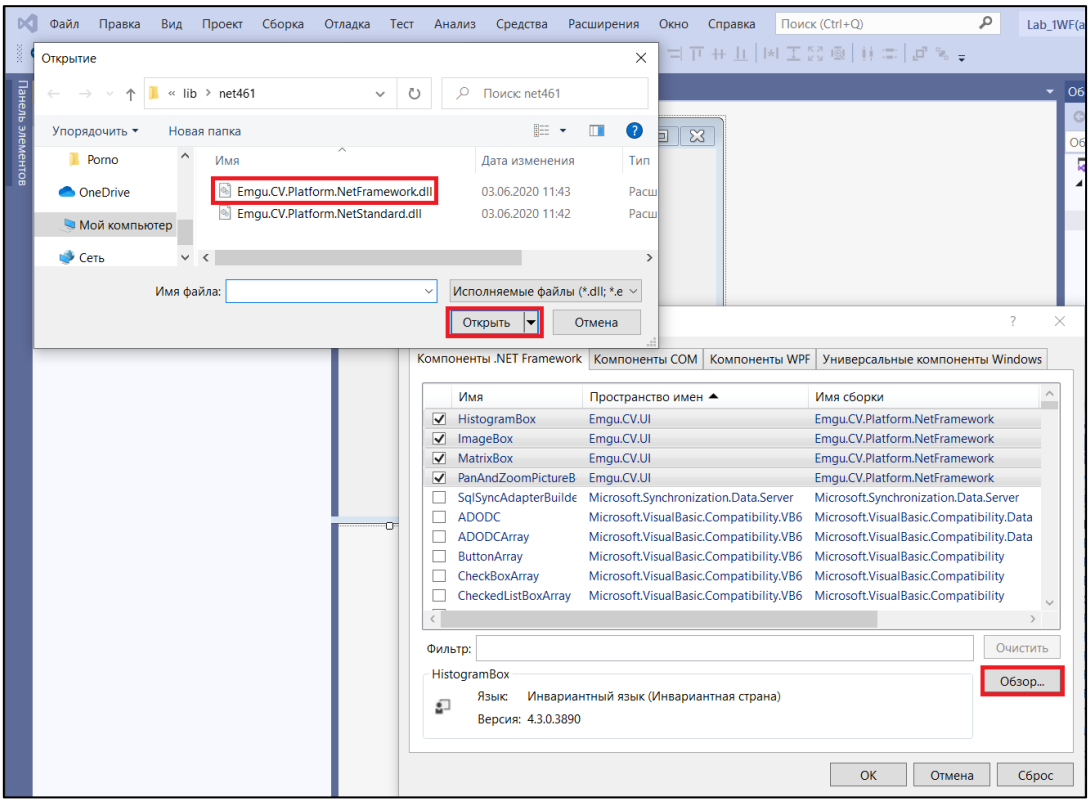


Для вызова кода загрузки изображения, можно использовать компонент типа Button.

В случае, если вкладка с компонентами Emgu.CV не появилась, вы можете добавить из самостоятельно. Для этого, кликните правой кнопкой мыши по панели элементов и выберите “Выбрать элементы…”:



В появившемся окне выберите “Обзор”, перейдите в директорию вашего проекта, затем в “`..\packages\Emgu.CV.4.3.0.3890\lib\net461\`”, и выберите dll файл как показано на изображении ниже:



В начале рабочего .cs-файла добавьте импорт необходимых библиотек:

```
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Emgu.CV.Util;
```

Работа с изображением:

Загрузка изображения с одновременным изменением его размера может быть реализована следующим образом:

```
private Image<Bgr, byte> sourceImage; //глобальная переменная

...

OpenFileDialog openFileDialog = new OpenFileDialog();
var result = openFileDialog.ShowDialog(); // открытие диалога выбора файла
if (result == DialogResult.OK) // открытие выбранного файла
{
       string fileName = openFileDialog.FileName;
       sourceImage = new Image<Bgr, byte>(fileName);
}
```

Отображения вашего изображения в Image можно осуществить с помощью поля Image:

```
imageBox1.Image = sourceImage.Resize(640, 480, Inter.Linear);
```

Метод `Resize` отвечает за изменение размеров изображения перед его выводом.

Преобразование цветового пространства изображения может быть осуществлено следующим образом (цветное -> оттенки серого):

```
Image<Gray, byte> grayImage = sourceImage.Convert<Gray, byte>();
```

Очистка преобразованного изображения от шумов выполняется следующим образом (изображение уменьшается -> увеличивается -> мелкие детали и шумы пропадают):

```
var tempImage = grayImage.PyrDown();
var destImage = tempImage.PyrUp();
```

Вызов фильтра Канни:

```
double cannyThreshold = 80.0;
double cannyThresholdLinking = 40.0;
Image<Gray, byte> cannyEdges = destImage.Canny(cannyThreshold, cannyThresholdLinking);
```

После вызова фильтра Канни, от изображения должны остаться только контуры объектов.

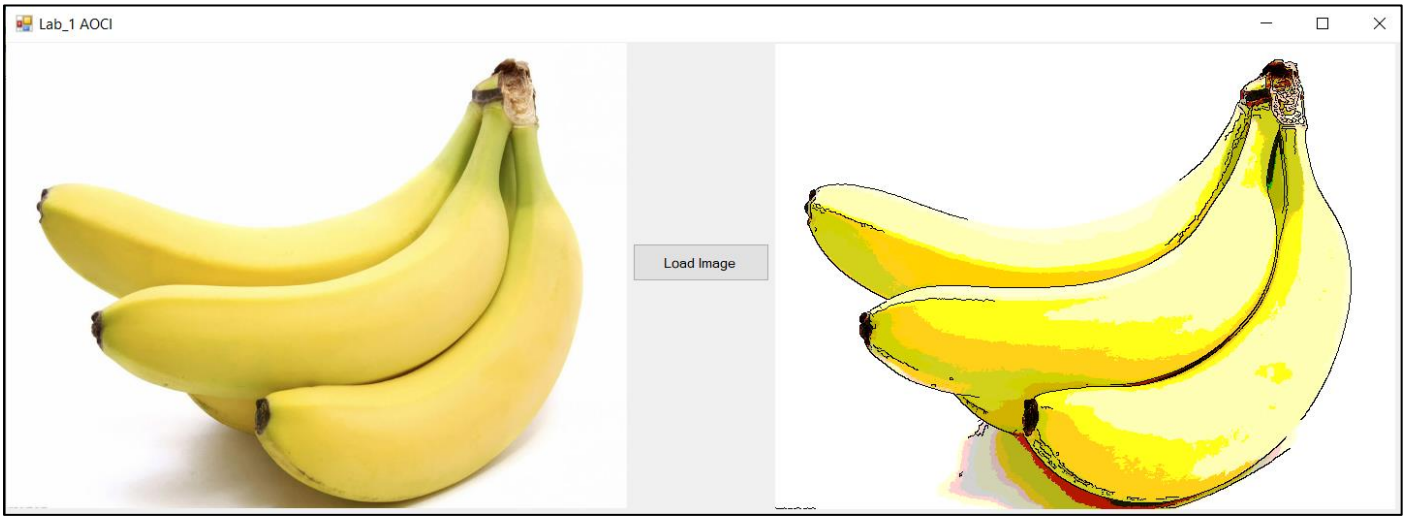
Реализация CellShading эффекта выглядит следующим образом:

```
var cannyEdgesBgr = cannyEdges.Convert<Bgr, byte>();
var resultImage = sourceImage.Sub(cannyEdgesBgr); // попиксельное вычитание

//обход по каналам
for (int channel = 0; channel < resultImage.NumberOfChannels; channel++)
       for (int x = 0; x < resultImage.Width; x++)
              for (int y = 0; y < resultImage.Height; y++) // обход по пискелям
              {
                     // получение цвета пикселя
                     byte color = resultImage.Data[y, x, channel];
                     if (color <= 50)
                         color = 0;
                     else if (color <= 100)
                         color = 25;
                     else if (color <= 150)
                         color = 180;
                     else if (color <= 200)
                         color = 210;
                     else
                         color = 255;

                    resultImage.Data[y, x, channel] = color; // изменение цвета пикселя
              }
```

Результат:

исходное изображение

обработанное изображение



Захват видео с веб-камеры:

Получить изображение с веб-камеры можно следующим образом:

```
private VideoCapture capture;
...
// инициализация веб-камеры
capture = new VideoCapture();
capture.ImageGrabbed += ProcessFrame;
capture.Start(); // начало обработки видеопотока
...
// захват кадра из видеопотока
private void ProcessFrame(object sender, EventArgs e)
{
    var frame = new Mat();
    capture.Retrieve(frame); // получение текущего кадра
}
...
capture.Stop(); // остановка обработки видеопотока
```

При необходимости, структуру типа Mat можно превратить в Image следующим образом:

```
Image<Bgr, byte> image = frame.ToImage<Bgr, byte>();
```

Захват видео из файла:

Захват видео из файла может быть выполнен аналогично, с учетом того, что при создании объекта класса VideoCapture необходимо передать строку с именем файла:

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

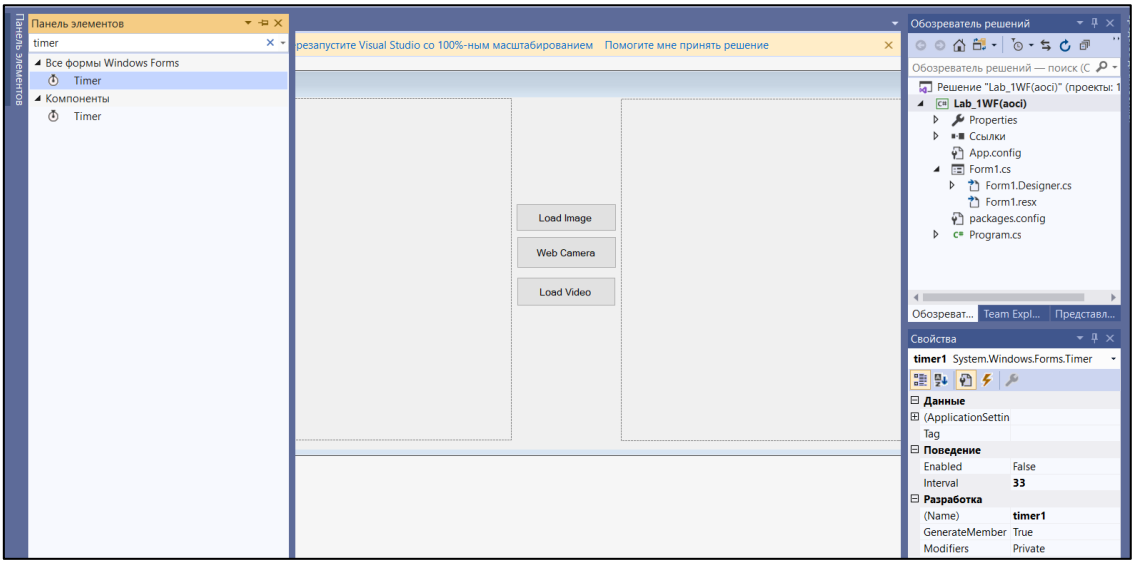
Захват кадра можно осуществлять при помощи метода:

```
var frame = capture.QueryFrame();
```

Получить информацию о видео можно прои помощи соответствующей функции и набора флагов:

```
var frameCount = capture.GetCaptureProperty(CapProp.FrameCount); //количество кадров в видео
var frameRate = capture.GetCaptureProperty(CapProp.XiFramerate); //кадроы в секунду в герцах
var frameHeight = capture.GetCaptureProperty(CapProp.FrameHeight); //высота кадров видео
var frameWidth = capture.GetCaptureProperty(CapProp.FrameWidth); //ширина кадров видео
```

Захват кадров видео лучше всего осуществлять по событию Tick компонента Timer:



Активировать таймер из кода можно при помощи выражения:

```
timer1.Enabled = true;
```

Интервал между вызовами события таймера можно задать в инспекторе объекта или из кода.

Задание:

Вывести на экран два изображения:

1. Результат работы фильтра Кэнни.
2. Результат работы порогового фильтра.
3. Добавить элементы интерфейса, позволяющие изменять: значения `cannyThreshold`, `cannyThresholdLinking` и пороговые значения для цветов.

Добавить в разрабатываемое приложение возможность захвата видеопотока с Web-камеры.

Добавить в разрабатываемое приложение возможность проигрывания видео файлов.

Добавить возможность применения разработанного фильтра к воспроизводимому видео.

Применить созданный фильтр к видео файлу.

Дополнительно: попробовать модифицировать пороговый фильтр, используя функции вместо пороговых значений.

Рекомендации:

1. Соблюдать codestyle (форматирование кода, наименование переменных, полей и методов и т.д.) для облегчения чтения кода и отладки.
2. Работу с изображением вынести в отдельный класс. В форме не должно быть никакой логики, кроме обработки событий из UI и вызовов соответствующих методов.
3. Использовать фильтр по расширению при выборе файлов. Пример: `openFileDialog.Filter = "Файлы изображений (*.jpg, *.jpeg, *.jpe, *.jfif, *.png) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png";`