Лабораторная работа №5:
Поиск текста и лиц на изображении

Цель:

Целью данной работы является изучение методик поиска текста и лиц на изображениях.

Задание:

Необходимо разработать приложение Windows Forms способное осуществлять:

1. Обнаружение и распознание текста.
2. Обнаружение лиц в видео потоке.

Поиск текста:

В упрощённом случае, текст, представляет собой сгруппированные участки изображения, имеющие отличный от фона цвет. Первым этапом обнаружения таких участков является преобразование изображения в монохромное, имеющее чёрный фон. В простейшем случае, для получения подобного изображения, достаточно порогового преобразования.

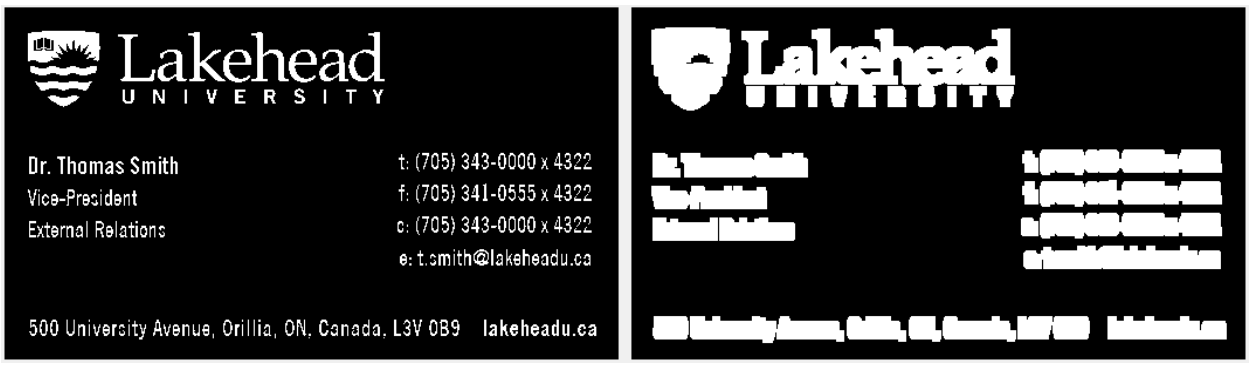
Результат преобразования:

Предполагается, что текст достаточно плотно сгруппирован, поэтому, если “расширить” отдельные буквы, то можно получить области, потенциально содержащие его.

Сделать это можно при помощи функции “расширение”:

```
var thresh = sourceImage.Convert<Gray, byte>();
thresh._ThresholdBinaryInv(new Gray(128), new Gray(255));
thresh._Dilate(5);
```

Результат преобразования:



Следующим этапом является выделение областей интереса для более детальной обработки. Осуществляется это путём нахождения контуров и вычисления их оганичивающих объёмов:

```
VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint();
CvInvoke.FindContours(thresh, contours, null, RetrType.List,ChainApproxMethod.ChainApproxSimple);

var output = sourceImage.Copy();

for (int i = 0; i < contours.Size; i++)
{
    if (CvInvoke.ContourArea(contours[i], false) > 50) //игнорирование маленьких контуров
    {
        Rectangle rect = CvInvoke.BoundingRectangle(contours[i]);
        output.Draw(rect, new Bgr(Color.Blue), 1);
    }
}
```

Результат работы:

После нахождения регионов, потенциально содержащих текст, их можно скопировать в виде отдельных изображений следующим образом:

```
input.ROI = rect; //установка региона интереса

Image<Bgr, byte> roiImg;
roiImg = sourceImage.Clone(); //копирование участка изображения, содержащего текст

sourceImage.ROI = System.Drawing.Rectangle.Empty;
```

А затем, использовать систему оптического распознавания символов.

В рамки данного курса не входит разработка подобных систем, поэтому предлагается использовать стандартный для Emgu CV OCR модуль. Подключить его можно добавив соответствующее пространство имён:

```
using Emgu.CV;
using Emgu.CV.Util;
using Emgu.CV.OCR; //модуль оптического распознавания символов
```

Для работы молуля распознования, необходимы данные обучения, которые можно скачать по адресу: https://github.com/tesseract-ocr/tessdata

По ссылке доступны наборы данных для разных языков. Все примеры в данной лабораторонй работе выполнены для английского языка.

Для использования модуля распознования, необходимо создать объект, указав путь к набору данных и язык:

```
Tesseract _ocr = new Tesseract("…\\tessdata", "eng", OcrEngineMode.TesseractLstmCombined);
```

Где “…\\tessdata” – путь к папке, в которую вы сохранили данные по ссылке выше. (путь не должен содержать имён на кирилице)

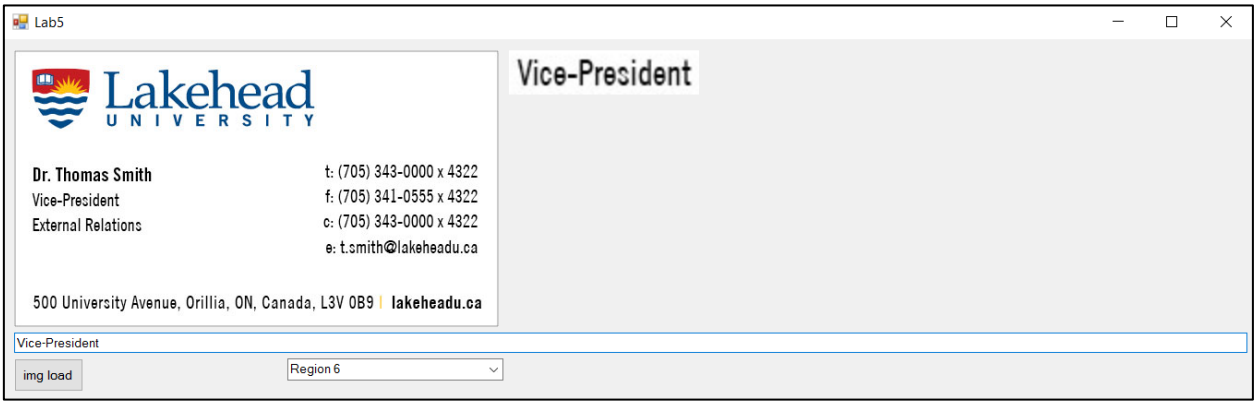
а затем, передать изображение содержащее текст:

```
_ocr.SetImage(roiImg); //фрагмент изображения, содержащий текст
_ocr.Recognize(); //распознание текста
Tesseract.Character[] words = _ocr.GetCharacters(); //получение найденных символов
```

получить результат распознания текста можно следующим образом:

```
StringBuilder strBuilder = new StringBuilder();
for (int i = 0; i < words.Length; i++)
{
    strBuilder.Append(words[i].Text);
}
```

Результат работы:



Возможные улучшения:

В случаях, если изображение помимо областей текста содержит другие контрастные элементы, после нахождения регионов интереса, имеет смысл провести дополнительную проверку. Каждая буква является отдельным контуром, а значит, области текста должны содержать в себе множество небольших контуров.

Получить иерархию контуров можно следующим образом:

```
Mat hierarchy = new Mat();
CvInvoke.FindContours(sourceImage,        outputResult,        hierarchy,        RetrType.Tree,
ChainApproxMethod.ChainApproxSimple);
```

Где:

`hierarchy[0, i, 0]` – индекс следующего контура на том же уровне иерархии (с тем же предком) или -1, если следующего контура не существует.

`hierarchy[0, i, 1]` – индекс предыдущего контура на том же уровне иерархии или -1.

`hierarchy[0, i, 2]` – индекс дочернего контура или -1.

`hierarchy[0, i, 3]` – индекс родительского контура или -1.

i – номер контура в массиве контуров.

После нахождения иерархии контуров, можно отбросить области, не содержащие в себе контура. Кроме того, иерархия контуров может быть полезна при определении, например, номерных знаков. Одним из признаков номерного знака может быть “контур, содержащий в себе 6 контуров”.

Обнаружение лиц:

Обнаружение лиц осуществляется при помощи поиска набора характерных признаков на изображении и установления соответствия их относительного взаиморасположения некоему шаблону. Шаблоны характерных признаков можно скачать по адресу: https://github.com/opencv/opencv/tree/master/data/haarcascades

Шаблон характерных признаков “лица” хранится в виде xml файла и может быть использован в вашем приложении следующим образом:

```
List<Rectangle> faces = new List<Rectangle>();

using (CascadeClassifier face = new
CascadeClassifier("…\\haarcascades\\haarcascade_frontalface_default.xml"))
{
    using (Mat ugray = new Mat())
    {
        CvInvoke.CvtColor(sourceImage, ugray, Emgu.CV.CvEnum.ColorConversion.Bgr2Gray);
        Rectangle[] facesDetected = face.DetectMultiScale(ugray,1.1,10,new Size(20,20));

        faces.AddRange(facesDetected);
    }
}
```

Где параметрами метода поиска лиц `DetectMultiScale` являются:

1) чёрно белый вариант целевого изображения
2) коэффициент масштабирования, должен быть больше 1.0, чем значение ближе к 1.0, тем больше чувствительность поиска и медленнее работа функции
3) минимальное число ближайших соседей, чем больше значение, тем меньше ложно позитивных срабатываний

Обозначить найденные на изображении лица можно при помощи цикла:

```
foreach (Rectangle rect in faces)
    input.Draw(rect, new Bgr(Color.Yellow), 2);
```

В данном примере, используется шаблон для поиска фронтального изображения лица, соответственно, для корректного срабатывания, желательно, что бы люди на фото стояли лицом к камере.

Результат работы:

Обнаружив области, потенциально содержащие лица, можно наложить на них изображение “маску”. Сделать это можно следующим образом:

```
foreach (Rectangle rect in faces) //для каждого лица
{
    res.ROI = rect; //для области содержащей лицо

    Image<Bgra, byte> small = frame.ToImage<Bgra, byte>().Resize(rect.Width, rect.Height,
Inter.Nearest); //создание

    //копирование изображения small на изображение res с использованием маски копирования mask
    CvInvoke.cvCopy(small, res, mask);

    res.ROI = System.Drawing.Rectangle.Empty;
}
```

Загрузить изображение с поддержкой прозрачности можно при помощи метода:

```
Mat frame = CvInvoke.Imread(f.FileName, ImreadModes.Unchanged);
```

Результат наложения маски:

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. Выделение участков изображения, потенциально содержащих текст.
2. Выбор и отображение участка изображения.
3. Получение текста, содержащегося на выбранном участке изображения.
4. Получение текста с видео потока.
5. Обнаружение и обозначение лиц на видео потоке.
6. Наложение “масок” на найденные в видеопотоке лица.