Название работы: Поиск текста и лиц на изображении

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

Постановка задачи:

Необходимо разработать приложение Windows Forms способное осуществлять:

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

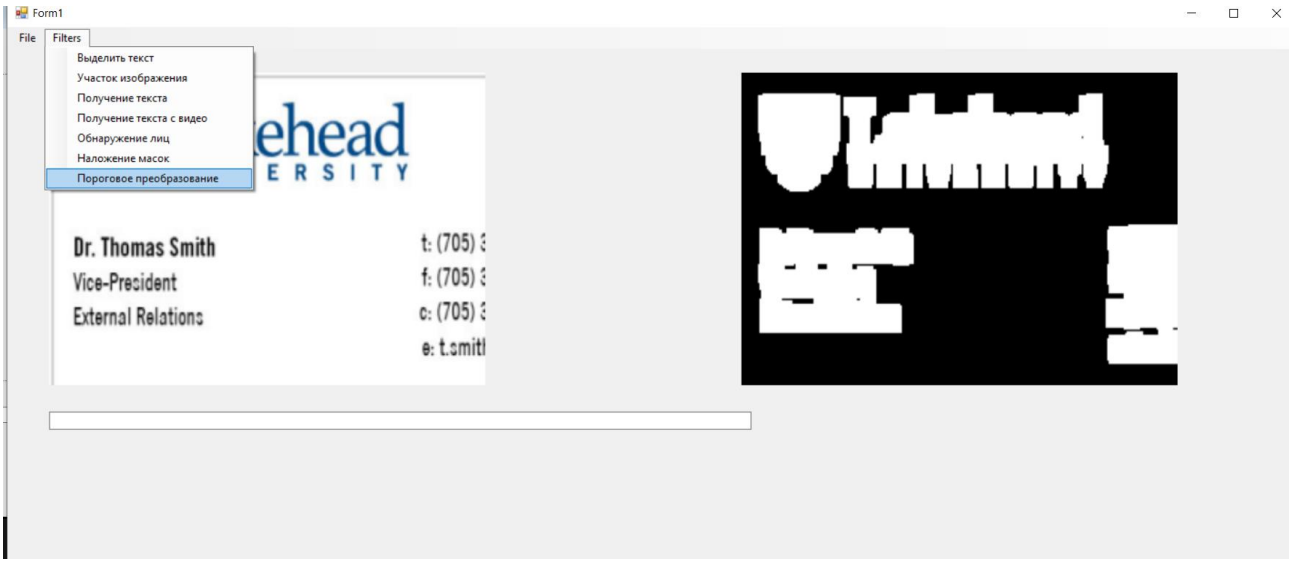
Задание:

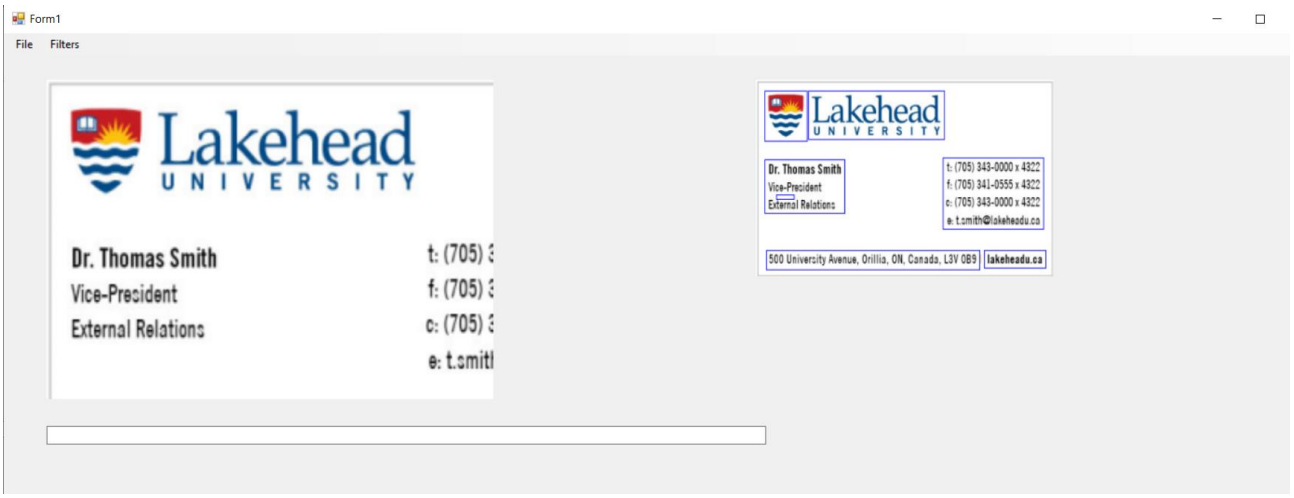
Реализовать программное средство, позволяющее отображать в одном окне два изображения,

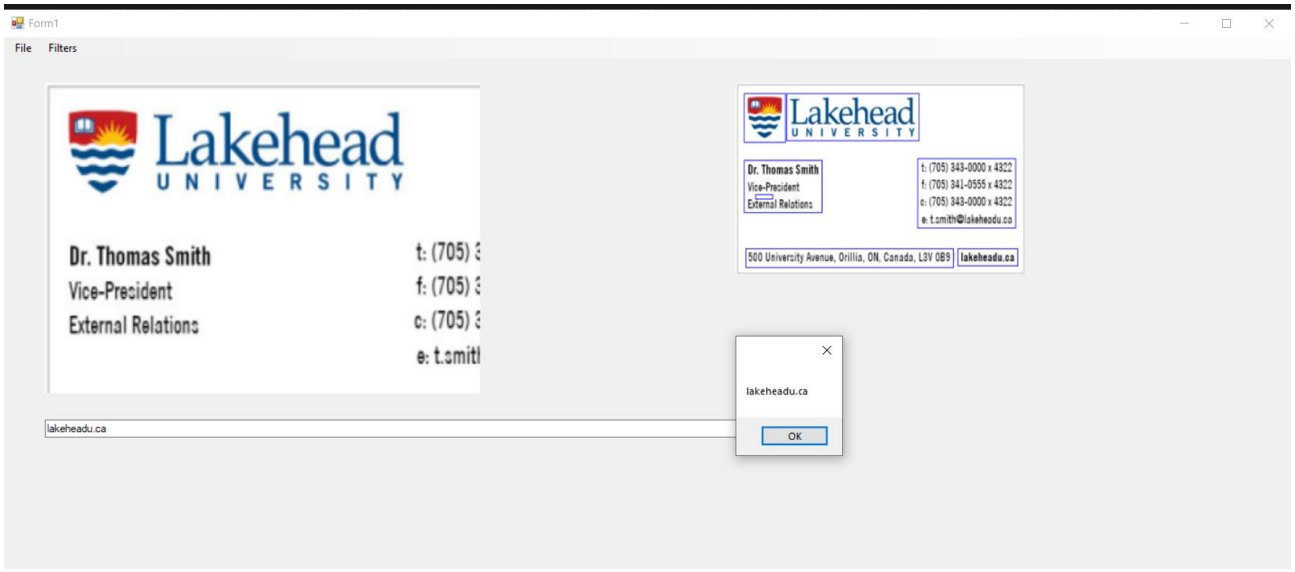
“оригинальное” слева и “результат обработки” справа. Реализовать интерфейс, позволяющий по

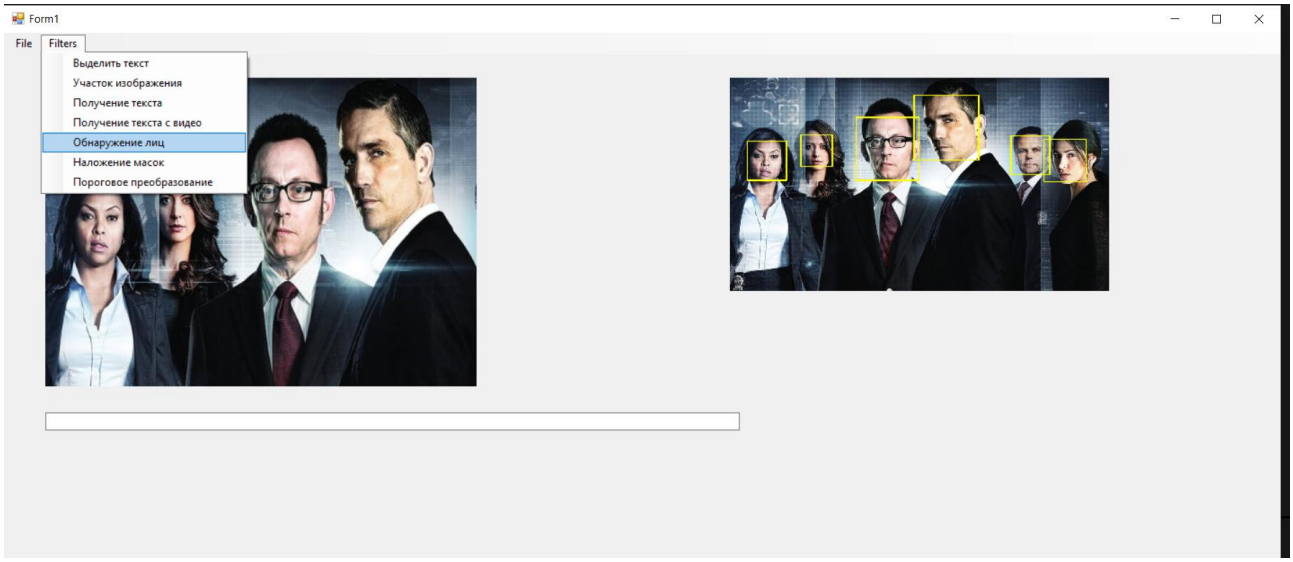
нажатию на соответствующие кнопки выполнять следующие операции:

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 лиц


Form1
File
Filters
Выделить текст
Участок изображения
Получение текста
Получение текста с видео
Обнаружение лиц
Наложение масок
Пороговое преобразование
Dr. Thomas Smith
Vice-President
External Relations


Form1
File
Filters
Lakehead
UNIVERSITY
Dr. Thomas Smith
Vice-President
External Relations
t: (705) 343-0000 x 4322
f: (705) 341-0555 x 4322
c: (705) 343-0000 x 4322
e: t.smith@lakeheadu.ca
500 University Avenue, Orillia, ON, Canada, L3V 0B9
lakeheadu.ca





Листинг кода:

```
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Emgu.CV;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using Emgu.CV.Util;
using Emgu.CV.OCR;
using Emgu.CV.UI;
using System.Security.Cryptography;

//1.Выделение участков изображения, потенциально содержащих текст. done
//2. Выбор и отображение участка изображения. done
//3. Получение текста, содержащегося на выбранном участке изображения. done
//4. Получение текста с видео потока. done
//5. Обнаружение и обозначение лиц на видео потоке. done
//6. Наложение “масок” на найденные в видеопотоке лица.

namespace Roviac_5
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private VideoCapture capture;
      bool Video = false;
      public Image<Bgr, byte> sourceImage { get; set; }
      private void openToolStripMenuItem_Click(object sender, EventArgs e)
      {
         OpenFileDialog ofd = new OpenFileDialog();
```

```
            ofd.Filter = "Picture Files (*jpg, *.png)| *.jpg;*.png";
            var result = ofd.ShowDialog();

            if (result == DialogResult.OK)
            {
                string filename = ofd.FileName;
                sourceImage = new Image<Bgr, byte>(filename);
                imageBox1.Image = sourceImage.Resize(640, 422, Inter.Linear);
            }
        }

        public static Image<Gray, byte> FindingAreasOfinterest(Image<Bgr, byte>
sourceImage)
        {
            var grayImage = sourceImage.Convert<Gray, byte>();
            var imgBinarize = sourceImage.Convert<Gray, byte>();

            CvInvoke.Threshold(grayImage, imgBinarize, 128, 255,
Emgu.CV.CvEnum.ThresholdType.BinaryInv);
            imgBinarize = imgBinarize.SmoothGaussian(1);
            imgBinarize._Dilate(5);
            return imgBinarize;
        }

        private void пороговоеПреобразованиеToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            imageBox2.Image = FindingAreasOfinterest(sourceImage).Resize(640, 422,
Inter.Linear);
        }
```

```
        public static Image<Bgr, byte> Highlight_text(Image<Bgr, byte> sourceImage)
        {
            var thresh = FindingAreasOfinterest(sourceImage);

            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);
                }
            }

            return output;
        }
        private void выделитьТекстToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            imageBox2.Image = Highlight_text(sourceImage).Resize(640, 422,
Inter.Linear);
        }
```

```
public static Image<Bgr, byte> roi_picture(Image<Bgr, byte> sourceImage, TextBox textBox1)
{
    Tesseract _ocr = new Tesseract("L:\\tessdata", "eng", OcrEngineMode.TesseractLstmCombined);
    var thresh = FindingAreasOfinterest(sourceImage);

    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);

          output.ROI = rect; //установка региона интереса
          Image<Bgr, byte> roiImg;
          roiImg = output.Clone(); //копирование участка изображения, содержащего текст
          output.ROI = System.Drawing.Rectangle.Empty;
          _ocr.SetImage(roiImg); //фрагмент изображения, содержащий текст
          _ocr.Recognize(); //распознание текста
          Tesseract.Character[] words = _ocr.GetCharacters(); //получение найденных символов
          StringBuilder strBuilder = new StringBuilder();
          for (int j = 0; j < words.Length; j++)
```

```
                {
                    strBuilder.Append(words[j].Text);
                }
                strBuilder.AppendLine();
                textBox1.Text = strBuilder.ToString();
                MessageBox.Show(textBox1.Text);
            }
        }
        return sourceImage;
    }

    public static Image<Bgr, byte> roi_picture2(Image<Bgr, byte> sourceImage,
TextBox textBox1)
    {
        Tesseract _ocr = new Tesseract("L:\\tessdata", "eng",
OcrEngineMode.TesseractLstmCombined);
        var thresh = FindingAreasOfinterest(sourceImage);

        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);
```

```
                output.ROI = rect; //установка региона интереса
                Image<Bgr, byte> roiImg;
                roiImg = output.Clone(); //копирование участка изображения,
содержащего текст
                output.ROI = System.Drawing.Rectangle.Empty;
                _ocr.SetImage(roiImg); //фрагмент изображения, содержащий текст
                _ocr.Recognize(); //распознание текста
                Tesseract.Character[] words = _ocr.GetCharacters(); //получение
найденных символов
                StringBuilder strBuilder = new StringBuilder();
                for (int j = 0; j < words.Length; j++)
                {
                   strBuilder.Append(words[j].Text);
                }
                strBuilder.AppendLine();
                //textBox1.Text = strBuilder.ToString();
                MessageBox.Show(strBuilder.ToString());
                //MessageBox.Show(textBox1.Text);
            }
        }
        return sourceImage;
    }
    private void получениеТекстаToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        imageBox2.Image = roi_picture(sourceImage, textBox1);
    }


    /// Обработка видево
```

```
        private void ProcessFrameVideo(object sender, EventArgs e)
        {
            var frame = new Mat();
            capture.Retrieve(frame); // получение текущего кадра
            imageBox1.Image = frame;
            var picture = frame.ToImage<Bgr, byte>();
            imageBox2.Image = roi_picture2(picture, textBox1);

Thread.Sleep((int)capture.GetCaptureProperty(Emgu.CV.CvEnum.CapProp.Fps));
        }
        private void openVideoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = "Video Files (*mp4, *.flv)| *.mp4;*.flv";
            var result = openFileDialog.ShowDialog(); // открытие диалога выбора
файла
            if (result == DialogResult.OK) // открытие выбранного файла
            {
                Video = true;
                string fileName = openFileDialog.FileName;
                capture = new VideoCapture(fileName);
                capture.ImageGrabbed += ProcessFrameVideo;
                capture.Start(); // начало обработки видеопотока
            }
        }

        private void получениеТекстаСВидеоToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            var frame = new Mat();
```

```
            capture.Retrieve(frame); // получение текущего кадра
            var picture = frame.ToImage<Bgr, byte>();
            imageBox1.Image = picture.Resize(640, 480, Inter.Linear);
            //imageBox2.Image = picture2.Resize(640, 480, Inter.Linear);
            //imageBox1.Image = frame.ToImage<Bgr, byte>();
            //imageBox2.Image = frame.ToImage<Gray,byte>();

Thread.Sleep((int)capture.GetCaptureProperty(Emgu.CV.CvEnum.CapProp.Fps));
        }

        public static Image<Bgr, byte> FindFaces(Image<Bgr, byte> sourceImage)
        {
            List<Rectangle> faces = new List<Rectangle>();
            using (CascadeClassifier face = new
            CascadeClassifier("L:\\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);
                }
            }
            foreach (Rectangle rect in faces)
                sourceImage.Draw(rect, new Bgr(Color.Yellow), 2);


            return sourceImage;
```

```
        }
        private void обнаружениеЛицToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            imageBox2.Image = FindFaces(sourceImage);
        }
        public static Image<Bgr, byte> FindFacesMask(Image<Bgr, byte> sourceImage)
        {
            OpenFileDialog f = new OpenFileDialog();
            f.Filter = "Picture Files (*jpg, *.png)| *.jpg;*.png";
            var result = f.ShowDialog();
            Image<Bgr, byte> sourceImage2 = sourceImage;
            if (result == DialogResult.OK)
            {
                string filename = f.FileName;
                sourceImage2 = new Image<Bgr, byte>(filename);
            }

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

            List<Rectangle> faces = new List<Rectangle>();
            using (CascadeClassifier face = new
            CascadeClassifier("L:\\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);
        }
    }
    foreach (Rectangle rect in faces)
        sourceImage.Draw(rect, new Bgr(Color.Yellow), 2);

    var mask = sourceImage2;
    foreach (Rectangle rect in faces) //для каждого лица
    {
        sourceImage.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
        try
        {
            CvInvoke.cvCopy(small, sourceImage, mask);
        }
        catch (Exception)
        {

        }
        //CvInvoke.cvCopy(small, sourceImage, mask);
        sourceImage.ROI = System.Drawing.Rectangle.Empty;
    }



    return sourceImage;
```

```
        }

        private void наложениеМасокToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            imageBox2.Image = FindFacesMask(sourceImage);
        }

        ////////////////////////////////////////////
    }
}
```