



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 5

Дисциплина: «Распознавание образов в информационных и автоматизированных системах копия 1»

Тема: «Поиск текста и лиц на изображении»

Выполнил: студент группы 211-723

Сергеев Станислав Олегович

Дата, подпись _____
(Дата) (Подпись)

Проверил: _____
(Фамилия И.О., степень, звание) (Оценка)

Дата, подпись _____
(Дата) (Подпись)

Замечания: _____

Москва

2022

Поиск текста и лиц на изображении.

Цель:

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

Постановка задачи:

Необходимо разработать приложение Windows Forms способное осуществлять:

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

Листинг программы

```
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 System.Text.RegularExpressions;
using Emgu.CV.UI;

namespace _5
{
    public partial class Form1 : Form
    {
        private Image<Bgr, byte> sourceImage;
        private VideoCapture capture;
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                OpenFileDialog ofd = new OpenFileDialog();
                ofd.Filter = "Изображения (*.jpg, *.jpeg, *.png) | *.jpg; *.jpeg; *.png";
                if (ofd.ShowDialog() == DialogResult.OK)
                {
                    sourceImage = new Image<Bgr, byte>(ofd.FileName);
                }
                imageBox1.Image = sourceImage.Resize(400, 400, Inter.Linear);
            }
            catch (Exception err)
            {
                MessageBox.Show(err.ToString());
            }
        }

        private void button2_Click(object sender, EventArgs e)
```

```

{
    var grayImage = advancedImageInteraction.grayscale(sourceImage.Clone());
    var contours = advancedImageInteraction.contoursFind(grayImage);
    var contoursImage = advancedImageInteraction.contoursDraw(sourceImage, contours);
    var ocrFromContours = advancedImageInteraction.ocr(sourceImage, contours);
    textBox1.Text = ocrFromContours;
    imageBox2.Image = contoursImage.Resize(400, 400, Inter.Linear);
}
private void button3_Click(object sender, EventArgs e)
{
    var facesImage = advancedImageInteraction.detectFaces(sourceImage);
    imageBox2.Image = facesImage.Resize(400, 400, Inter.Linear);
}
private void button4_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        capture = new VideoCapture(openFileDialog.FileName);
    }
    capture.ImageGrabbed += ProcessFrame;
}
private void ProcessFrame(object sender, EventArgs e)
{
    try
    {
        var frame = new Mat();
        capture.Retrieve(frame);
        Image<Bgr, byte> frameImage = frame.ToImage<Bgr, byte>().Resize(400, 400,
Inter.Linear);
        imageBox2.Image = advancedImageInteraction.detectFaces(frameImage);
        Thread.Sleep((int)capture.GetCaptureProperty(Emgu.CV.CvEnum.CapProp.Fps));
    }
    catch (Exception)
    {
    }
}
private void button5_Click(object sender, EventArgs e)
{
    capture.Start();
}
private void button6_Click(object sender, EventArgs e)
{
    capture.Pause();
}
}
public partial class advancedImageInteraction
{
    public static Image<Gray, byte> grayscale(Image<Bgr, byte> _sourceImage)
    {
        var grayImage = _sourceImage.Convert<Gray, byte>();
        grayImage._ThresholdBinaryInv(new Gray(128), new Gray(255));
        grayImage._Dilate(5);
        return grayImage;
    }
    public static VectorOfVectorOfPoint contoursFind(Image<Gray, byte> grayImg)
    {
        VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint();
        CvInvoke.FindContours(grayImg, contours, null, RetrType.List,
ChainApproxMethod.ChainApproxSimple);
        return contours;
    }
    public static Image<Bgr, byte> contoursDraw(Image<Bgr, byte> _sourceImage,
VectorOfVectorOfPoint contours)
    {
        var dstImg = _sourceImage.Copy();
        for (int i = 0; i < contours.Size; i++)
        {
            if (CvInvoke.ContourArea(contours[i], false) > 50)

```

```

        {
            Rectangle rect = CvInvoke.BoundingRectangle(contours[i]);
            dstImg.Draw(rect, new Bgr(Color.Blue), 1);
        }
    }
    return dstImg;
}

public static String ocr(Image<Bgr, byte> _sourceImage, VectorOfVectorOfPoint contours)
{
    StringBuilder strBuilder = new StringBuilder();
    var dstImg = _sourceImage.Copy();
    for (int i = 0; i < contours.Size; i++)
    {
        if (CvInvoke.ContourArea(contours[i], false) > 50)
        {
            Rectangle rect = CvInvoke.BoundingRectangle(contours[i]);
            _sourceImage.ROI = rect;
            Image<Bgr, byte> roiImg;
            roiImg = _sourceImage.Clone();
            _sourceImage.ROI = System.Drawing.Rectangle.Empty;
            Tesseract _ocr = new Tesseract("D:\\tessdata-main",
"eng", OcrEngineMode.TesseractLstmCombined);
            _ocr.SetImage(roiImg);
            _ocr.Recognize();
            Tesseract.Character[] words = _ocr.GetCharacters();
            for (; i < words.Length; i++)
            {
                strBuilder.Append(words[i].Text);
            }
        }
    }
    return strBuilder.ToString();
}

public static Image<Bgr, byte> detectFaces(Image<Bgr, byte> _sourceImage)
{
    Image<Bgr, byte> dstImg = _sourceImage.Clone();
    List<Rectangle> faces = new List<Rectangle>();
    using (CascadeClassifier face = new
CascadeClassifier("D:\\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);
        }
    }
    foreach (Rectangle rect in faces) dstImg.Draw(rect, new Bgr(Color.AliceBlue), 3);
    return dstImg;
}
}
}

```



