عنوان: OpenCVSharp #18
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۰۴ ۱۰:۵۰
آدرس: www.dotnettips.info
گروه‌ها: OCR, OpenCV

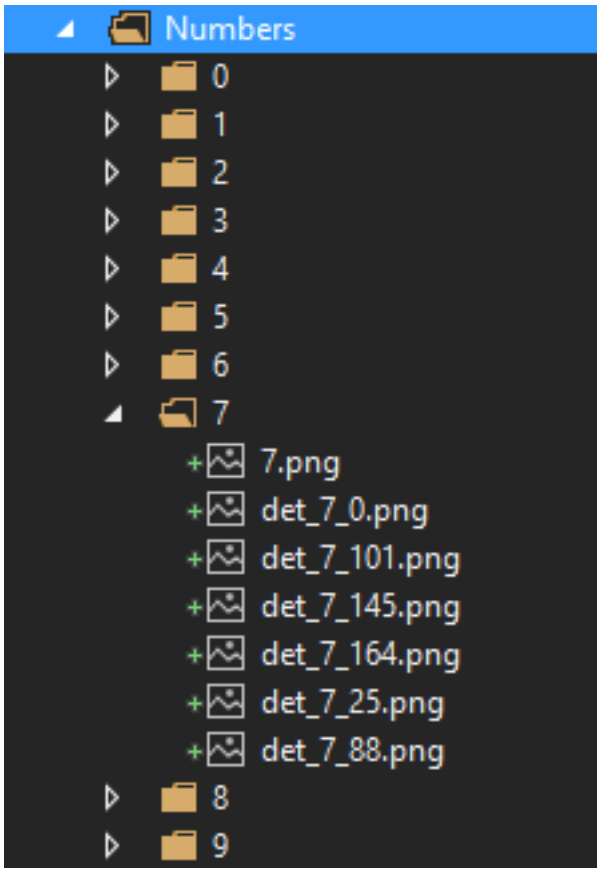
ساخت یک OCR ساده تشخیص اعداد انگلیسی به کمک OpenCV

این مطلب را می‌توان به عنوان جمع بندی مطالبی که تاکنون بررسی شدند درنظر گرفت و در اساس مطلب جدیدی ندارد و صرفا ترکیب یک سری تکنیک است؛ برای مثال:

چطور یک تصویر را به نمونه‌ی سیاه و سفید آن تبدیل کنیم؟
کار با متد Threshold جهت بهبود کیفیت یک تصویر جهت تشخیص اشیاء
تشخیص کانتورها (Contours) و اشیاء موجود در یک تصویر
آشنایی با نحوه‌ی گروه بندی تصاویر مشابه و مفاهیمی مانند برچسب‌های تصاویر که بیانگر یک گروه از تصاویر هستند.

تهیه تصاویر اعداد انگلیسی جهت آموزش دادن به الگوریتم CvKNearest

در اینجا نیز از یکی دیگر از الگوریتم‌های machine learning موجود در OpenCV به نام CvKNearest برای تشخیص اعداد انگلیسی استفاده خواهیم کرد. این الگوریتم نزدیک‌ترین همسایه‌ی اطلاعاتی مفروض را در گروهی از داده‌های آموزش داده شده‌ی به آن پیدا می‌کند. خروجی آن شماره‌ی این گروه است. بنابراین نحوه‌ی طبقه‌ی بندی اطلاعات در اینجا چیزی شبیه به شکل زیر خواهد بود:



مجموعه‌ای از تصاویر 0 تا 9 را جمع آوری کرده‌ایم. هر کدام از پوشه‌ها، بیانگر اعدادی از یک خانواده هستند. این تصویر را با

فرمت ذیل جمع آوری می‌کنیم:

```
public class ImageInfo
{
    public Mat Image { set; get; }
    public int ImageGroupId { set; get; }
    public int ImageId { set; get; }
}
```

به این ترتیب

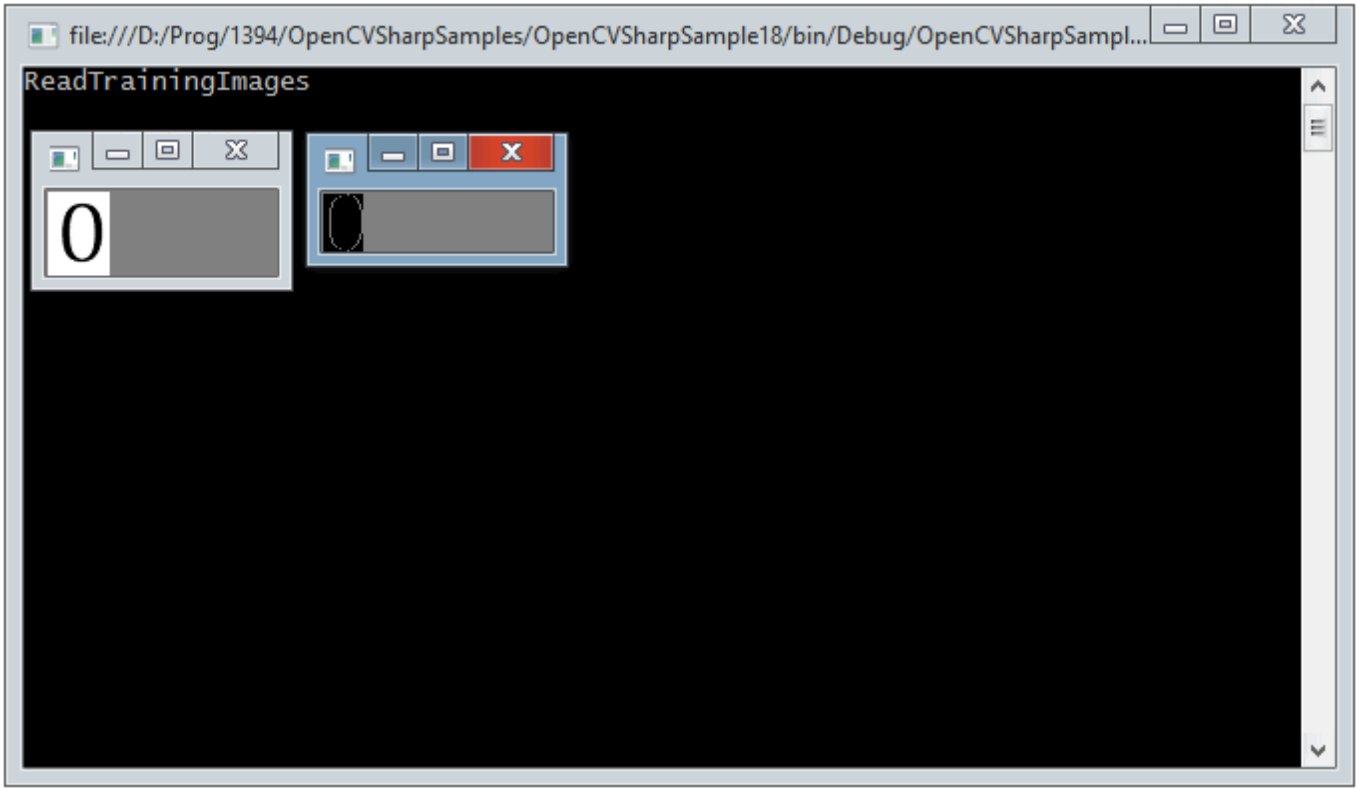
```
public IList<ImageInfo> ReadTrainingImages(string path, string ext)
{
    var images = new List<ImageInfo>();

    var imageId = 1;
    foreach (var dir in new DirectoryInfo(path).GetDirectories())
    {
        var groupId = int.Parse(dir.Name);
        foreach (var imageFile in dir.GetFiles(ext))
        {
            var image = processTrainingImage(new Mat(imageFile.FullName, LoadMode.GrayScale));
            if (image == null)
            {
                continue;
            }

            images.Add(new ImageInfo
            {
                Image = image,
                ImageId = imageId++,
                ImageGroupId = groupId
            });
        }
    }

    return images;
}
```

در متد خواندن تصاویر آموزشی، ابتدا پوشه‌های اصلی مسیر Numbers تصویر ابتدای بحث دریافت می‌شوند. سپس نام هر پوشه، شماره‌ی گروه تصاویر موجود در آن پوشه را تشکیل خواهد داد. به این نام در الگوریتم‌های machine leaning، کلاس هم گفته می‌شود. سپس هر تصویر را با فرمت سیاه و سفید بارگذاری کرده و به لیست تصاویر موجود اضافه می‌کنیم. در اینجا از متد processTrainingImage نیز استفاده شده‌است. هدف از آن بهبود کیفیت تصویر دریافتی جهت کار تشخیص اشیاء است:

```
private static Mat processTrainingImage(Mat gray)
{
    var threshImage = new Mat();
    Cv2.Threshold(gray, threshImage, Thresh, ThresholdMaxVal, ThresholdType.BinaryInv); // Threshold to
find contour

    Point[][] contours;
    HiearchyIndex[] hierarchyIndexes;
    Cv2.FindContours(
        threshImage,
        out contours,
        out hierarchyIndexes,
        mode: ContourRetrieval.CComp,
        method: ContourChain.ApproxSimple);

    if (contours.Length == 0)
    {
        return null;
    }

    Mat result = null;

    var contourIndex = 0;
    while ((contourIndex >= 0))
    {
        var contour = contours[contourIndex];

        var boundingRect = Cv2.BoundingRect(contour); //Find bounding rect for each contour
        var roi = new Mat(threshImage, boundingRect); //Crop the image
```

```
        //Cv2.ImShow("src", gray);
        //Cv2.ImShow("roi", roi);
        //Cv.WaitKey(0);

        var resizedImage = new Mat();
        var resizedImageFloat = new Mat();
        Cv2.Resize(roi, resizedImage, new Size(10, 10)); //resize to 10X10
        resizedImage.ConvertTo(resizedImageFloat, MatType.CV_32FC1); //convert to float
        result = resizedImageFloat.Reshape(1, 1);

        contourIndex = hierarchyIndexes[contourIndex].Next;
    }

    return result;
}
```

عملیات صورت گرفته‌ی در این متد را با تصویر ذیل بهتر می‌توان توضیح داد:



ابتدا تصویر اصلی بارگذاری می‌شود؛ همان تصویر سمت چپ. سپس با استفاده از متد Threshold، شدت نور نواحی مختلف آن یکسان شده و آماده می‌شود برای تشخیص کانتورهای موجود در آن. در ادامه با استفاده از متد FindContours، شیء مرتبط با عدد جاری یافت می‌شود. سپس متد Cv2.BoundingRect مستطیل دربرگیرنده‌ی این شیء را تشخیص می‌دهد (تصویر سمت راست). بر این اساس می‌توان تصویر اصلی ورودی را به یک تصویر کوچکتر که صرفا شامل ناحیه‌ی عدد مدنظر است، تبدیل کرد.
در ادامه برای کار با الگوریتم CvKNearest نیاز است تا این تصویر بهبود یافته را تبدیل به یک ماتریس یک بعدی کردی که روش انجام کار توسط متد Reshape مشاهده می‌کنید.
از همین روش پردازش و بهبود تصویر ورودی، جهت پردازش اعداد یافت شده‌ی در یک تصویر با تعداد زیادی عدد نیز استفاده خواهیم کرد.

آموزش دادن به الگوریتم CvKNearest

تا اینجا تصاویر گروه بندی شده‌ای را خوانده و لیستی از آن‌ها را مطابق فرمت الگوریتم CvKNearest تهیه کردیم. مرحله‌ی بعد، معرفی این لیست به متد Train این الگوریتم است:

```
public CvKNearest TrainData(IList<ImageInfo> trainingImages)
{
    var samples = new Mat();
    foreach (var trainingImage in trainingImages)
    {
        samples.PushBack(trainingImage.Image);
    }

    var labels = trainingImages.Select(x => x.ImageGroupId).ToArray();
    var responses = new Mat(labels.Length, 1, MatType.CV_32SC1, labels);
    var tmp = responses.Reshape(1, 1); //make continuous
    var responseFloat = new Mat();
    tmp.ConvertTo(responseFloat, MatType.CV_32FC1); // Convert  to float


    var kNearest = new CvKNearest();
    kNearest.Train(samples, responseFloat); // Train with sample and responses
    return kNearest;
}
```

متد Train دو ورودی دارد. ورودی اول آن یک تصویر است که باید از طریق متد PushBack کلاس Mat تهیه شود. بنابراین لیست تصاویر اصلی را تبدیل به لیستی از Matها خواهیم کرد.
سپس نیاز است لیست گروه‌های متناظر با تصاویر اعداد را تبدیل به فرمت مورد انتظار متد Train کنیم. در اینجا صرفا لیستی از اعداد صحیح را داریم. این لیست نیز باید تبدیل به یک Mat شود که روش انجام آن در متد فوق بیان شده‌است. کلاس Mat سازنده‌ی مخصوصی را جهت تبدیل لیست اعداد، به همراه دارد. این Mat نیز باید تبدیل به یک ماتریس یک بعدی شود که برای این منظور از متد Reshape استفاده شده‌است.

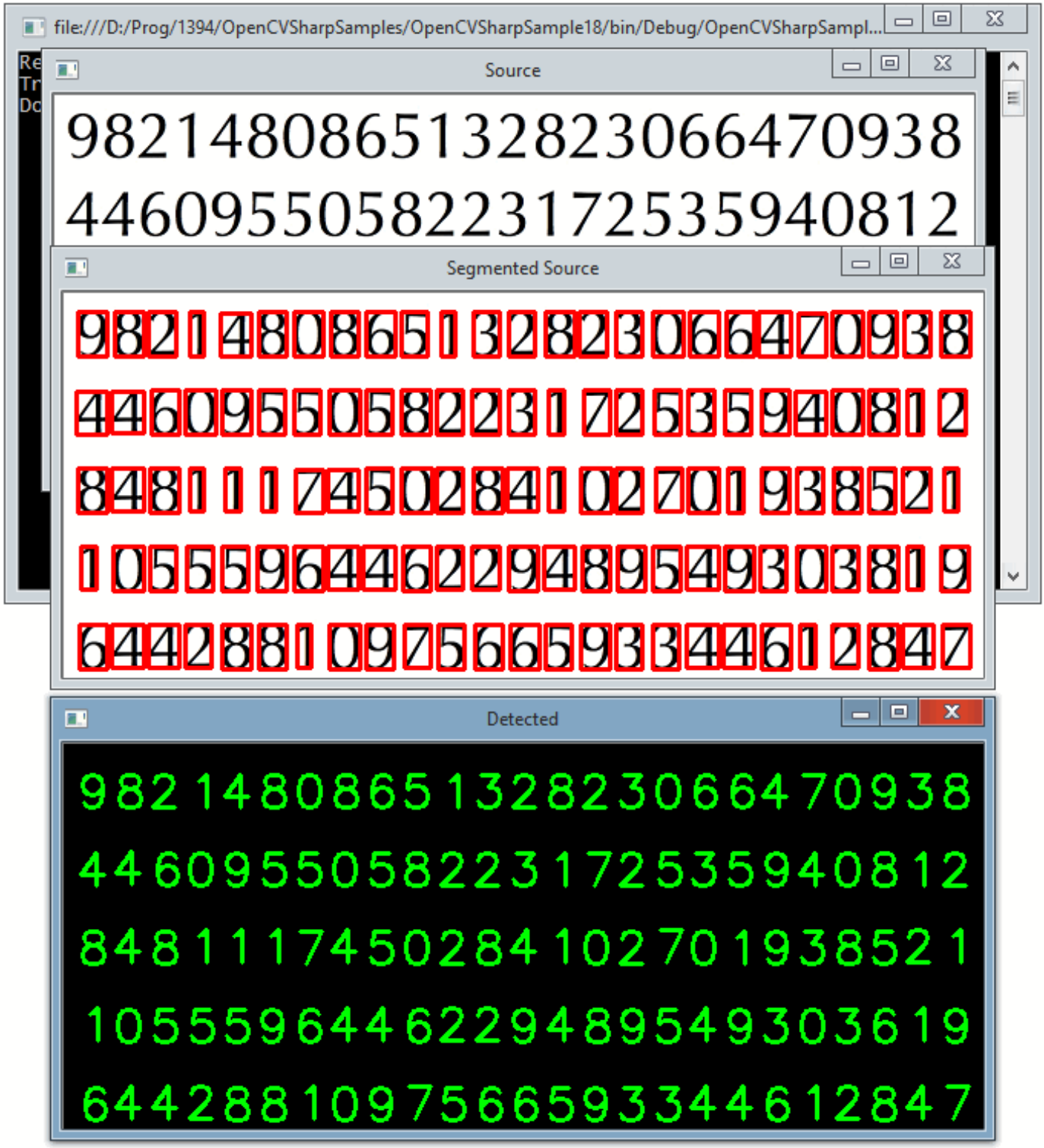
انجام عملیات OCR نهایی

پس از تهیه‌ی لیستی از تصاویر و آموزش دادن آن‌ها به الگوریتم CvKNearest، تنها کاری که باید انجام دهیم، یافتن اعداد در تصویر نمونه‌ی مدنظر و سپس معرفی آن به متد FindNearest الگوریتم CvKNearest است. روش انجام اینکار بسیار شبیه است به روش معرفی شده در متد processTrainingImage که پیشتر بررسی شد:

```
public void DoOCR(CvKNearest kNearest, string path)
{
    var src = Cv2.ImRead(path);
    Cv2.ImShow("Source", src);

    var gray = new Mat();
    Cv2.CvtColor(src, gray, ColorConversion.BgrToGray);

    var threshImage = new Mat();
    Cv2.Threshold(gray, threshImage, Thresh, ThresholdMaxVal, ThresholdType.BinaryInv); // Threshold to
find contour


    Point[][] contours;
    HiearchyIndex[] hierarchyIndexes;
    Cv2.FindContours(
        threshImage,
        out contours,
        out hierarchyIndexes,
        mode: ContourRetrieval.CComp,
        method: ContourChain.ApproxSimple);

    if (contours.Length == 0)
    {
        throw new NotSupportedException("Couldn't find any object in the image.");
    }

    //Create input sample by contour finding and cropping
    var dst = new Mat(src.Rows, src.Cols, MatType.CV_8UC3, Scalar.All(0));

    var contourIndex = 0;
```

```
    while ((contourIndex >= 0))
    {
        var contour = contours[contourIndex];

        var boundingRect = Cv2.BoundingRect(contour); //Find bounding rect for each contour

        Cv2.Rectangle(src,
            new Point(boundingRect.X, boundingRect.Y),
            new Point(boundingRect.X + boundingRect.Width, boundingRect.Y + boundingRect.Height),
            new Scalar(0, 0, 255),
            2);

        var roi = new Mat(threshImage, boundingRect); //Crop the image

        var resizedImage = new Mat();
        var resizedImageFloat = new Mat();
        Cv2.Resize(roi, resizedImage, new Size(10, 10)); //resize to 10X10
        resizedImage.ConvertTo(resizedImageFloat, MatType.CV_32FC1); //convert to float
        var result = resizedImageFloat.Reshape(1, 1);


        var results = new Mat();
        var neighborResponses = new Mat();
        var dists = new Mat();
        var detectedClass = (int)kNearest.FindNearest(result, 1, results, neighborResponses, dists);

        //Console.WriteLine("DetectedClass: {0}", detectedClass);
        //Cv2.ImShow("roi", roi);
        //Cv.WaitKey(0);

        //Cv2.ImWrite(string.Format("det_{0}_{1}.png",detectedClass, contourIndex), roi);

        Cv2.PutText(
            dst,
            detectedClass.ToString(CultureInfo.InvariantCulture),
            new Point(boundingRect.X, boundingRect.Y + boundingRect.Height),
            0,
            1,
            new Scalar(0, 255, 0),
            2);

        contourIndex = hierarchyIndexes[contourIndex].Next;
    }

    Cv2.ImShow("Segmented Source", src);
    Cv2.ImShow("Detected", dst);

    Cv2.ImWrite("dest.jpg", dst);

    Cv2.WaitKey();
}
```

این عملیات به صورت خلاصه در تصویر ذیل مشخص شده‌است:



ابتدا تصویر اصلی که قرار است عملیات OCR روی آن صورت گیرد، بارگذاری می‌شود. سپس کانتورها و اعداد موجود در آن تشخیص داده می‌شوند. مستطیل‌های قرمز رنگ در برگیرنده‌ی این اعداد را در تصویر دوم مشاهده می‌کنید. سپس این کانتورهای یافت شده را که شامل یکی از اعداد تشخیص داده شده‌است، تبدیل به یک ماتریس یک بعدی کرده و به متد FindNearest ارسال می‌کنیم. خروجی آن نام گروه یا پوشه‌ای است که این عدد در آن قرار دارد. در همینجا این خروجی را تبدیل به یک رشته کرده و در تصویر سوم با رنگ سبز رنگ نمایش می‌دهیم.

بنابراین در این تصویر، پنجره‌ی segmented image، همان اشیاء تشخیص داده شده‌ی از تصویر اصلی هستند.

پنجره‌ی با زمینه‌ی سیاه رنگ، نتیجه‌ی نهایی OCR است که نسبتا هم دقیق عمل کرده‌است.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید.

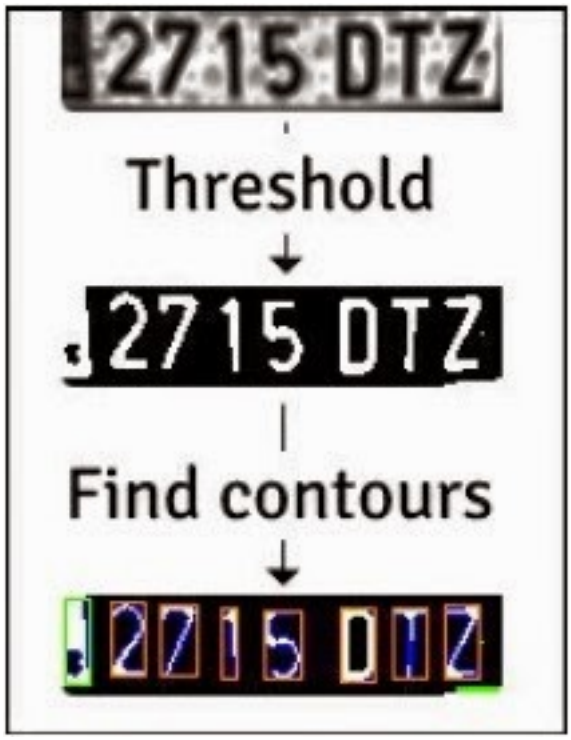
نظرات خوانندگان

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۴/۰۴/۰۵ ۱۵:۴۰

امکانش هست که با استفاده از این نکات، بتوان حروف پلاک یک ماشین را تشخیص داد؟
برای مثال از تک تک حروف ممکن برای یک پلاک، عکس گرفته و این روش را برای آن پیاده سازی کرد و یا کلا روش پیاده سازی آن متفاوت است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۰۵ ۱۶:۶

- مرحله‌ی اول، بیرون کشیدن مستطیل شماره پلاک خودرو از داخل یک عکس کلی است؛ چیزی شبیه به مطلب « تشخیص چهره ». اگر به پوشه‌ی دیتا OpenCV مراجعه کنید، فایل xml تشخیص مستطیل شماره پلاک خودروهای روسی را دارد؛ فایل‌های haarcascade_licence_plate_rus_16stages.xml و haarcascade_russian_plate_number.xml. نحوه‌ی استفاده‌ی از این فایل‌ها، دقیقا همانند مطلب تشخیص چهره‌است. برای تشخیص شماره پلاک‌های ایرانی، باید از روش کلی مطرح شده در مطلب « طراحی classifier سفارشی تشخیص خودروها » استفاده کنید. یک سری عکس تهیه کنید و بعد فایل XML آن‌را استخراج کنید.
- مرحله‌ی دوم، با مطلب جاری تفاوتی ندارد:



ابتدا اصل پلاک باید تشخیص داده شود (همان مطلب تشخیص چهره با یک فایل XML مناسب). بعد بهبود کیفیت تصویر پلاک و آماده سازی آن برای استخراج کانتورها است. سپس این اشیاء یافت شده را به الگوریتم مثلا CvKNearest ارسال و شماره‌ی گروه هر کانتور را دریافت می‌کنید (روش OCR مطلب جاری).

یک نکته‌ی تکمیلی

فایل‌های XML یافتن مستطیل شماره پلاک‌های چند کشور مختلف را در پروژه‌ی openalpr می‌توانید پیدا کنید. این پروژه از OpenCV برای تشخیص پلاک و سپس از Tesseract OCR برای انجام کار OCR نهایی استفاده می‌کند (Tesseract OCR یک OCR سورس باز تهیه شده توسط گوگل است).

نویسنده: امیران
تاریخ: ۱۳۹۴/۰۴/۰۷ ۹:۵۰

او سی آر tesseract از موتور leptonica برای پردازش تصاویر استفاده می‌کند. opencv معروفتر است. بنچمارکی برای مقایسه وجود دارد؟
در مقاله عنوان کردید برای بهبود کیفیت از threshold استفاده می‌کنیم در مقالات قبلی در همین زمینه بحثی راجع به morphology داشتید آیا راه حل نهایی ترکیبی از این دو است؟ مثلا برای متون خطی قدیمی ماشین تحریر با کیفیت پائین می‌توان از ترکیب این دو استفاده نمود؟
یکی از معضلات حل نشده در زمینه ocr فارسی، متون دست نویس است. راه حلی برای آن با استفاده از سلسله مطالب جاری می‌توان یافت یا حداقل مسیری برای حل آن؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۰۷ ۱۰:۰

مسلما یک برنامه‌ی OCR قوی باید دارای قسمتی به نام کالیبره کردن باشد و در اینجا می‌توان انواع و اقسام الگوریتم‌ها را برای رسیدن به بهترین نتیجه ترکیب کرد. برای مثال در مطلب فوق اگر پارامترهای متد threshold را تغییر دهید، دقت OCR متفاوت خواهد بود.
در پروژه‌ی نهایی بحث جاری، یک پوشه‌ی اعداد دست نویس انگلیسی هم هست که از آن می‌توان برای آموزش دادن به الگوریتم‌های machine learning مطرح شده استفاده کرد.

نویسنده: محسن نجف زاده
تاریخ: ۱۳۹۴/۰۴/۰۷ ۱۹:۳۱

مجموعه داده بزرگ HODA شامل ارقام فارسی که توسط دانشگاه تربیت مدرس ایجاده شده از وب سایت فارسی او سی آر قابل دریافت است .

- مشخصات و روند جمع آوری این مجموعه داده در سال 2007 میلادی در مجله Pattern Recognition Letters منتشر شد
- این مجموعه شامل 102,352 نمونه عدد فارسی است که از 12,000 فرم مربوط به آزمون ورودی کاردانی به کارشناسی و کارشناسی ارشد جمع آوری شده است.