عنوان: **ساخت یک بارکدخوان با استفاده از OpenCV و ZXing.Net**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۱۴ ۱۰:۳۵
آدرس: www.dotnettips.info
گروه‌ها: OpenCV, Barcodes

فرض کنید می‌خواهیم بارکد این قبض را یافته و سپس عدد متناظر با آن‌را در برنامه بخوانیم.


مراحل کار به این صورت هستند:

**بارگذاری تصویر و چرخش آن در صورت نیاز**

ابتدا تصویر بارکد دار را بارگذاری کرده و آن‌را تبدیل به یک تصویر سیاه و سفید می‌کنیم:

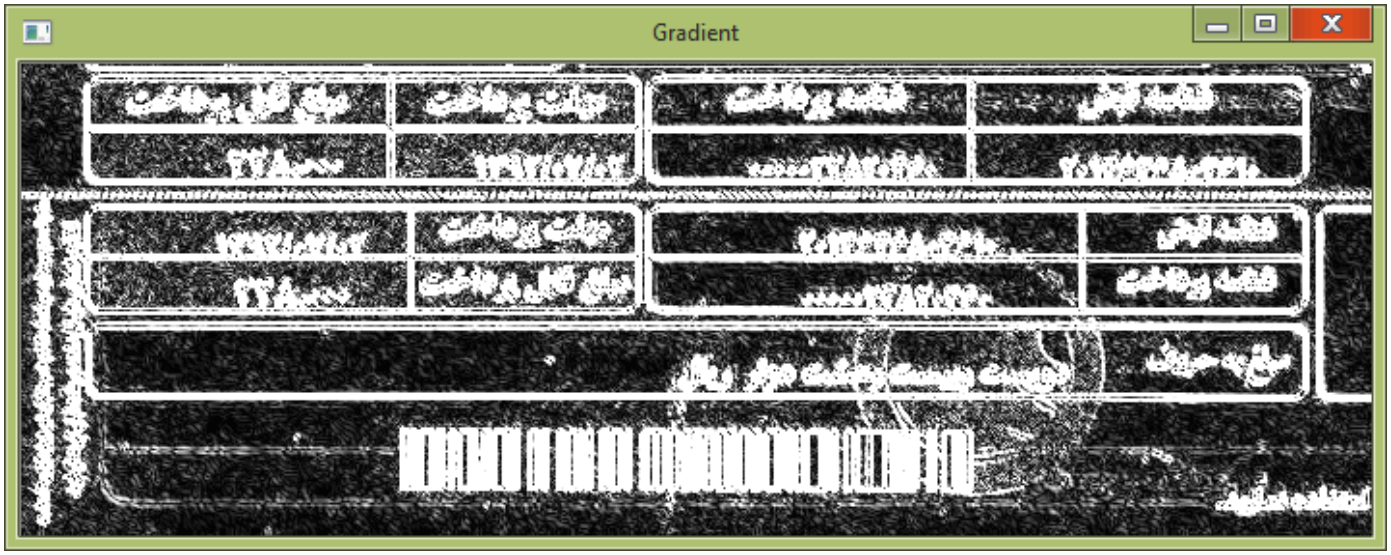
```
// load the image and convert it to grayscale
var image = new Mat(fileName);

if (rotation != 0)
{
    rotateImage(image, image, rotation, 1);
}

if (debug)
{
    Cv2.ImShow("Source", image);
    Cv2.WaitKey(1); // do events
}

var gray = new Mat();
var channels = image.Channels();
if (channels > 1)
{
    Cv2.CvtColor(image, gray, ColorConversion.BgrToGray);
}
else
{
    image.CopyTo(gray);
}
```

در این بین ممکن است بارکد موجود در تصویر، دقیقا در زاویه‌ای که در تصویر ابتدای بحث قرار گرفته‌است، وجود نداشته باشد؛ مثلا منهای 90 درجه، چرخیده باشد. به همین جهت می‌توان از متد چرخش تصویر مطلب « تغییر اندازه، و چرخش تصاویر » ارائه شده در قسمت نهم این سری استفاده کرد.

### تشخیص گرادیان‌های افقی و عمودی

یکی از روش‌های تشخیص بارکد، استفاده از روشی است که در تشخیص خودرو قسمت 16 بیان شد. تعداد زیادی تصویر بارکد را تهیه و سپس آن‌ها را به الگوریتم‌های machine learning جهت تشخیص و یافتن محدوده‌ی بارکد موجود در یک تصویر، ارسال کنیم. هرچند این روش جواب خواهد داد، اما در این مورد خاص، قسمت بارکد، شبیه به گرادیانی از رنگ‌ها است. کتابخانه‌ی OpenCV برای یافتن این نوع گرادیان‌ها دارای متدی است به نام Sobel :

```
// compute the Scharr gradient magnitude representation of the images
// in both the x and y direction
var gradX = new Mat();
Cv2.Sobel(gray, gradX, MatType.CV_32F, xorder: 1, yorder: 0, ksize: -1);
//Cv2.Scharr(gray, gradX, MatType.CV_32F, xorder: 1, yorder: 0);

var gradY = new Mat();
Cv2.Sobel(gray, gradY, MatType.CV_32F, xorder: 0, yorder: 1, ksize: -1);
//Cv2.Scharr(gray, gradY, MatType.CV_32F, xorder: 0, yorder: 1);

// subtract the y-gradient from the x-gradient
var gradient = new Mat();
Cv2.Subtract(gradX, gradY, gradient);
Cv2.ConvertScaleAbs(gradient, gradient);

if (debug)
{
    Cv2.ImShow("Gradient", gradient);
    Cv2.WaitKey(1); // do events
}
```




ابتدا درجه‌ی شدت گرادیان‌ها در جهت‌های x و y محاسبه می‌شوند. سپس این شدت‌ها از هم کم خواهند شد تا بیشترین شدت گرادیان موجود در محور x حاصل شود. این بیشترین شدت‌ها، بیانگر نواحی خواهند بود که احتمال وجود بارکدهای افقی در آن‌ها بیشتر است.

### کاهش نویز و یکی کردن نواحی تشخیص داده شده

در ادامه می‌خواهیم با استفاده از متدهای تشخیص کانتور ( قسمت 12 )، نواحی با بیشترین شدت گرادیان افقی را پیدا کنیم. اما تصویر حاصل از قسمت قبل برای اینکار مناسب نیست. به همین جهت با استفاده از متدهای کار با مورفولوژی تصاویر، این نواحی

گرادیانی را یکی می‌کنیم ( قسمت 8 ).

```
// blur and threshold the image
var blurred = new Mat();
Cv2.Blur(gradient, blurred, new Size(9, 9));

var threshImage = new Mat();
Cv2.Threshold(blurred, threshImage, thresh, 255, ThresholdType.Binary);

if (debug)
{
    Cv2.ImShow("Thresh", threshImage);
    Cv2.WaitKey(1); // do events
}


// construct a closing kernel and apply it to the thresholded image
var kernel = Cv2.GetStructuringElement(StructuringElementShape.Rect, new Size(21, 7));
var closed = new Mat();
Cv2.MorphologyEx(threshImage, closed, MorphologyOperation.Close, kernel);

if (debug)
{
    Cv2.ImShow("Closed", closed);
    Cv2.WaitKey(1); // do events
}


// perform a series of erosions and dilations
Cv2.Erode(closed, closed, null, iterations: 4);
Cv2.Dilate(closed, closed, null, iterations: 4);

if (debug)
{
    Cv2.ImShow("Erode & Dilate", closed);
    Cv2.WaitKey(1); // do events
}
```

این سه مرحله را در تصاویر ذیل مشاهده می‌کنید:


ابتدا با استفاده از متد Threshold، تصویر را به یک تصویر باینری تبدیل خواهیم کرد. در این تصویر تمام نقاط دارای شدت رنگ کمتر از مقدار thresh، به مقدار حداکثر 255 تنظیم می‌شوند.

سپس با استفاده از متدهای تغییر مورفولوژی تصویر، قسمت‌های مجاور به هم را می‌بندیم و یکی می‌کنیم. این مورد در یافتن اشیاء احتمالی که ممکن است بارکد باشند، بسیار مفید است.

متدهای Erode و Dilate در اینجا کار حذف نویزهای اضافی را انجام می‌دهند؛ تا بهتر بتوان بر روی نواحی بزرگتر یافت شده، تمرکز کرد.

**یافتن بزرگترین ناحیه‌ی به هم پیوسته‌ی موجود در یک تصویر**

تمام این مراحل را انجام دادیم تا بتوانیم بزرگترین ناحیه‌ی به هم پیوسته‌ای را که احتمال می‌رود بارکد باشد، در تصویر تشخیص دهیم. پس از این آماده سازی‌ها، اکنون با استفاده از متد یافتن کانتورها، تمام نواحی یکی شده را یافته و بزرگترین مساحت ممکن را به عنوان بارکد انتخاب می‌کنیم:

```
//find the contours in the thresholded image, then sort the contours
//by their area, keeping only the largest one

Point[][] contours;
HiearchyIndex[] hierarchyIndexes;
Cv2.FindContours(
    closed,
    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.");
}

var contourIndex = 0;
var previousArea = 0;
var biggestContourRect = Cv2.BoundingRect(contours[0]);
while ((contourIndex >= 0))
{
    var contour = contours[contourIndex];

    var boundingRect = Cv2.BoundingRect(contour); //Find bounding rect for each contour
    var boundingRectArea = boundingRect.Width * boundingRect.Height;
    if (boundingRectArea > previousArea)
    {
        biggestContourRect = boundingRect;
        previousArea = boundingRectArea;
    }

    contourIndex = hierarchyIndexes[contourIndex].Next;
}


var barcode = new Mat(image, biggestContourRect); //Crop the image
Cv2.CvtColor(barcode, barcode, ColorConversion.BgrToGray);

Cv2.ImShow("Barcode", barcode);
Cv2.WaitKey(1); // do events
```

حاصل این عملیات یافتن بزرگترین ناحیه‌ی گرادیانی به هم پیوسته‌ی موجود در تصویر است:



**خواندن مقدار متناظر با بارکد یافت شده**

خوب، تا اینجا موفق شدیم، محل قرارگیری بارکد را تصویر پیدا کنیم. مرحله‌ی بعد خواندن مقدار متناظر با این تصویر است. برای این منظور از کتابخانه‌ی سورس بازی به نام http://zxingnet.codeplex.com استفاده خواهیم کرد. این کتابخانه قادر است بارکد بسازد و همچنین تصاویر بارکدها را خوانده و مقادیر متناظر با آن‌ها را استخراج کند. برای نصب آن می‌توان از دستور ذیل استفاده کرد:

```
PM> Install-Package ZXing.Net
```

پس از نصب این کتابخانه‌ی بارکدساز و بارکد خوان، اکنون تنها کاری که باید صورت گیرد، ارسال تصویر بارکد جدا شده‌ی توسط OpenCV به آن است:

```
private static string getBarcodeText(Mat barcode)
{
    // `ZXing.Net` needs a white space around the barcode
    var barcodeWithWhiteSpace = new Mat(new Size(barcode.Width + 30, barcode.Height + 30),
MatType.CV_8U, Scalar.White);
    var drawingRect = new Rect(new Point(15, 15), new Size(barcode.Width, barcode.Height));
    var roi = barcodeWithWhiteSpace[drawingRect];
    barcode.CopyTo(roi);

    Cv2.ImShow("Enhanced Barcode", barcodeWithWhiteSpace);
    Cv2.WaitKey(1); // do events

    return decodeBarcodeText(barcodeWithWhiteSpace.ToBitmap());
}

private static string decodeBarcodeText(System.Drawing.Bitmap barcodeBitmap)
{
    var source = new BitmapLuminanceSource(barcodeBitmap);

    // using http://zxingnet.codeplex.com/
    // PM> Install-Package ZXing.Net
    var reader = new BarcodeReader(null, null, ls => new GlobalHistogramBinarizer(ls))
    {
        AutoRotate = true,
        TryInverted = true,
        Options = new DecodingOptions
```

```
        {
            TryHarder = true,
            //PureBarcode = true,
            /*PossibleFormats = new List<BarcodeFormat>
                    {
                        BarcodeFormat.CODE_128
                        //BarcodeFormat.EAN_8,
                        //BarcodeFormat.CODE_39,
                        //BarcodeFormat.UPC_A
                    }*/
        }
    };

    //var newhint = new KeyValuePair<DecodeHintType, object>(DecodeHintType.ALLOWED_EAN_EXTENSIONS, new
Object());
    //reader.Options.Hints.Add(newhint);

    var result = reader.Decode(source);
    if (result == null)
    {
        Console.WriteLine("Decode failed.");
        return string.Empty;
    }

    Console.WriteLine("BarcodeFormat: {0}", result.BarcodeFormat);
    Console.WriteLine("Result: {0}", result.Text);


    var writer = new BarcodeWriter
    {
        Format = result.BarcodeFormat,
        Options = { Width = 200, Height = 50, Margin = 4},
        Renderer = new ZXing.Rendering.BitmapRenderer()
    };
    var barcodeImage = writer.Write(result.Text);
    Cv2.ImShow("BarcodeWriter", barcodeImage.ToMat());

    return result.Text;
}
```

چند نکته را باید در مورد کار با ZXing.Net بخاطر داشت؛ وگرنه جواب نمی‌گیرید:

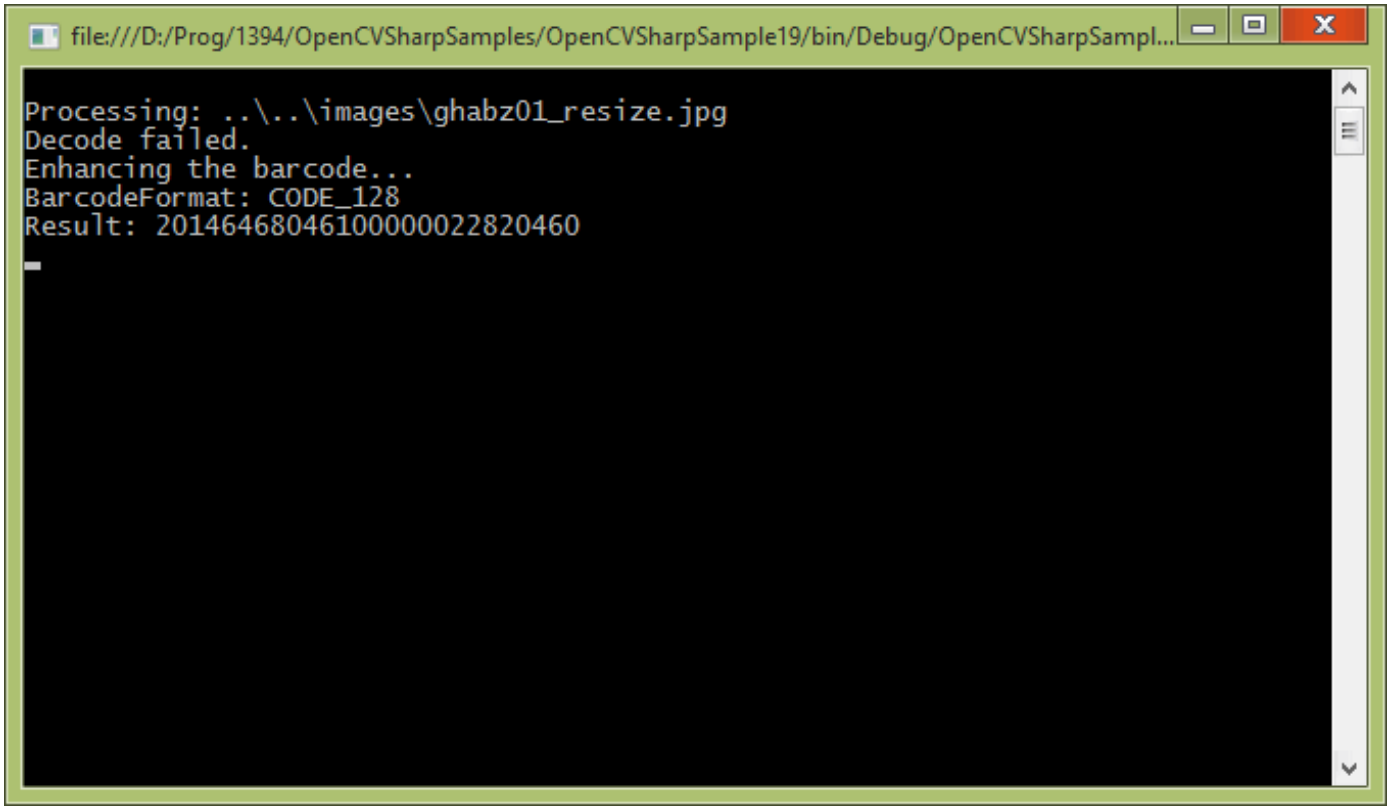
الف) این کتابخانه حتما نیاز دارد تا تصویر بارکد، در یک حاشیه‌ی سفید در اختیار او قرار گیرد. به همین جهت در متد getBarcodeText، ابتدا تصویر بارکد یافت شده، به میانه‌ی یک مستطیل سفید رنگ بزرگ‌تر کپی می‌شود.

ب) برای تبدیل Mat به Bitmap مورد نیاز این کتابخانه می‌توان از متد الحاقی ToBitmap استفاده کرد ( قسمت 7 ).

ج) پس از آن وهله‌ای از کلاس BarcodeReader آماده شده و در آن پارامترهایی مانند بیشتر سعی کن (TryHarder) و اصلاح درجه‌ی چرخش تصویر (AutoRotate) تنظیم شده‌اند.

د) بارکدهای موجود در قبض‌های ایران عموما بر اساس فرمت CODE_128 ساخته می‌شوند. بنابراین برای خواندن سریعتر آنها می‌توان PossibleFormats را مقدار دهی کرد. اگر این مقدار دهی صورت نگیرد، تمام حالت‌های ممکن بررسی می‌شوند.

در آخر کار این متد، از متد Writer آن نیز برای تولید بارکد مشابهی استفاده شده‌است تا بتوان بررسی کرد این دو تا چه اندازه به هم شبیه هستند.



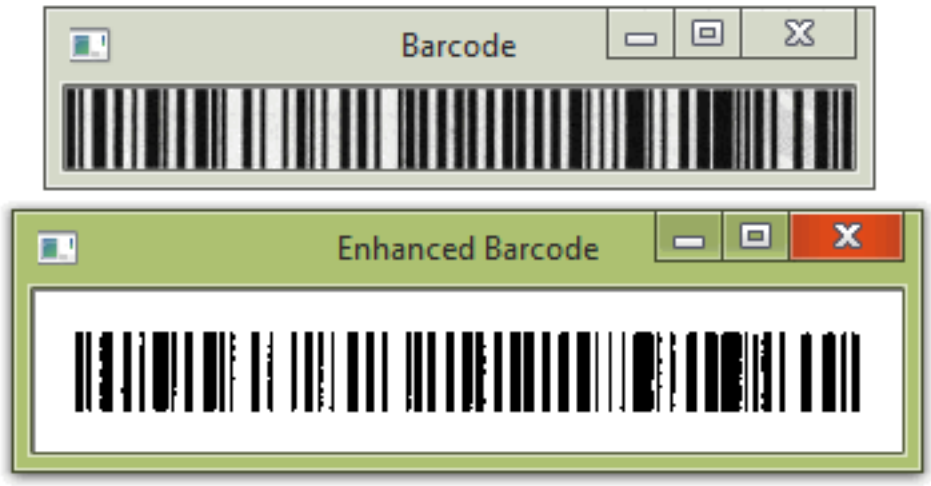

همانطور که مشاهده می‌کنید، عدد تشخیص داده شده، با عدد شناسه‌ی قبض و شناسه‌ی پرداخت تصویر ابتدای بحث یکی است.

**بهبود تصویر، پیش از ارسال آن به متد Decode کتابخانه‌ی ZXing.Net**

در تصویر قبلی، سطر decode failed را هم ملاحظه می‌کنید. علت اینجا است که اولین سعی انجام شده، موفق نبوده است؛ چون تصویر تشخیص داده شده، بیش از اندازه نویز و حاشیه‌ی خاکستری دارد. می‌توان این حاشیه‌ی خاکستری را با دوبار اعمال متد Threshold از بین برد:

```
var barcodeClone = barcode.Clone();
var barcodeText = getBarcodeText(barcodeClone);

if (string.IsNullOrWhiteSpace(barcodeText))
{
    Console.WriteLine("Enhancing the barcode...");
    //Cv2.AdaptiveThreshold(barcode, barcode, 255,
        //AdaptiveThresholdType.GaussianC, ThresholdType.Binary, 9, 1);
    //var th = 119;
    var th = 100;
    Cv2.Threshold(barcode, barcode, th, 255, ThresholdType.ToZero);
    Cv2.Threshold(barcode, barcode, th, 255, ThresholdType.Binary);
    barcodeText = getBarcodeText(barcode);
}

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

if (debug)
{
    Cv2.ImShow("Segmented Source", image);
    Cv2.WaitKey(1); // do events
}

Cv2.WaitKey(0);
```

```
Cv2.DestroyAllWindows();
```




اعداد یافت شده، دقیقا از روی تصویر بهبود یافته‌ی توسط متدهای Threshold خوانده شده‌اند و نه تصویر ابتدایی یافت شده. بنابراین به این موضوع نیز باید دقت داشت.

**کدهای کامل این مثال را** از اینجا **می‌توانید دریافت کنید.**