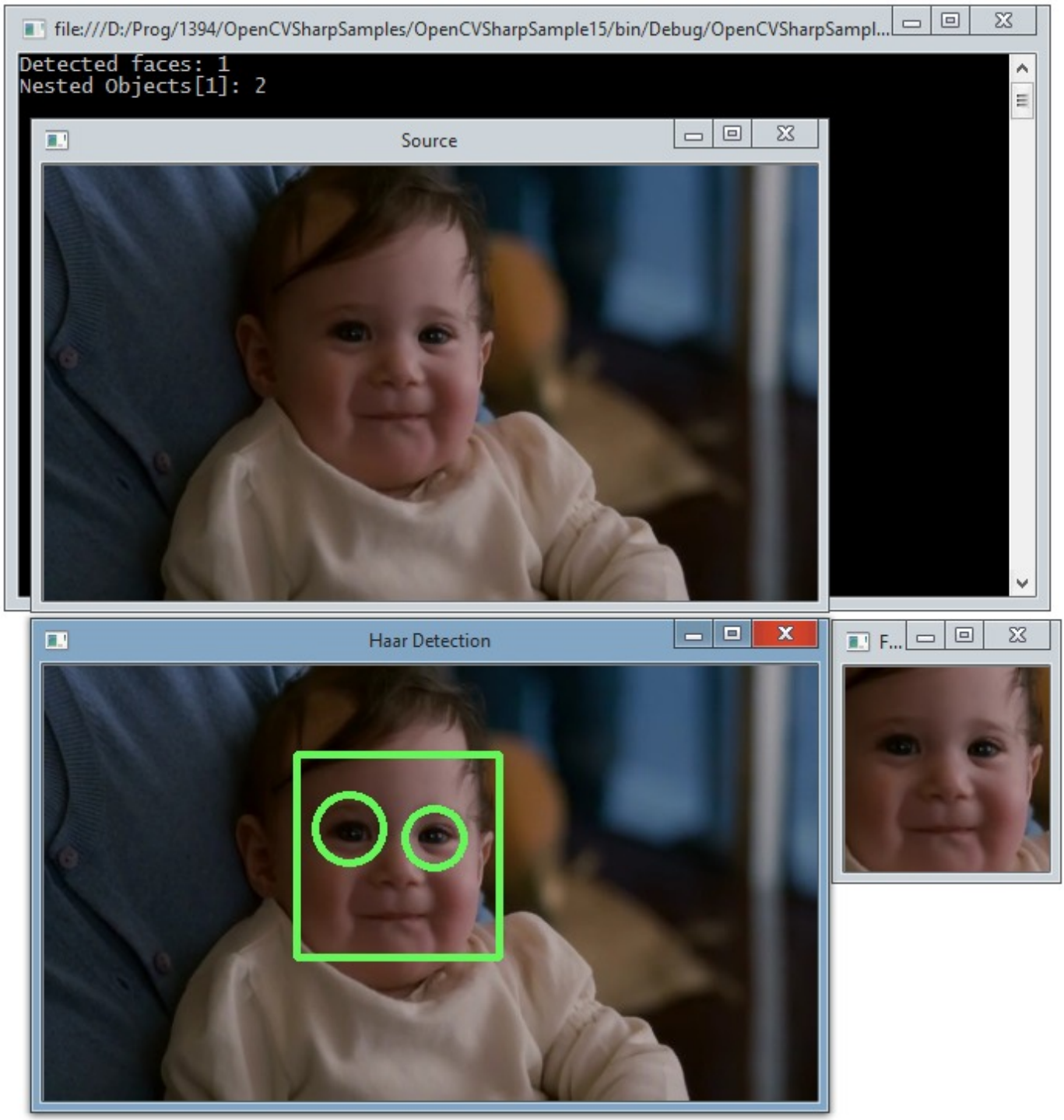
عنوان: OpenCVSharp #15
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۲۹ ۱۵:۲۰
آدرس: www.dotnettips.info
گروه‌ها: OpenCV

تشخیص چهره به کمک OpenCV

OpenCV به کمک الگوریتم‌های machine learning (در اینجا Haar feature-based cascade classifiers) و داده‌های مرتبط با آن‌ها، قادر است اشیاء پیچیده‌ای را در تصاویر پیدا کند. برای پیگیری مثال بحث جاری نیاز است کتابخانه‌ی اصلی OpenCV را دریافت کنید؛ از این جهت که به فایل‌های XML موجود در پوشه‌ی opencv\sources\data\haarcascades آن نیاز داریم. در اینجا از دو فایل haarcascade_eye_tree_eyeglasses.xml و haarcascade_frontalface_alt.xml آن استفاده خواهیم کرد (این دوفایل جهت سهولت کار، به همراه مثال این بحث نیز ارائه شده‌اند).

فایل haarcascade_frontalface_alt.xml اصطلاحا trained data مخصوص یافتن چهره‌ی انسان در تصاویر است و فایل haarcascade_eye_tree_eyeglasses.xml کمک می‌کند تا بتوان در چهره‌ی یافت شده، چشمان شخص را نیز با دقت بالایی تشخیص داد؛ چیزی همانند تصویر ذیل:



مراحل تشخیص چهره توسط OpenCVSharp

ابتدا همانند سایر مثال‌های OpenCV، تصویر مدنظر را به کمک کلاس Mat بارگذاری می‌کنیم:

```
var srcImage = new Mat(@"..\..\Images\Test.jpg");
Cv2.ImShow("Source", srcImage);
Cv2.WaitKey(1); // do events

var grayImage = new Mat();
Cv2.CvtColor(srcImage, grayImage, ColorConversion.BgrToGray);
Cv2.EqualizeHist(grayImage, grayImage);
```

همچنین در اینجا جهت بالا رفتن سرعت کار و بهبود دقت تشخیص چهره، این تصویر اصلی به یک تصویر سیاه و سفید تبدیل شده‌است و سپس متد Cv2.EqualizeHist بر روی آن فراخوانی گشته‌است. این متد وضوح تصویر را جهت یافتن الگوها بهبود می‌بخشد.
سپس فایل xml یاد شده‌ی در ابتدای بحث را توسط کلاس CascadeClassifier بارگذاری می‌کنیم:

```
var cascade = new CascadeClassifier(@"..\..\Data\haarcascade_frontalface_alt.xml");
var nestedCascade = new CascadeClassifier(@"..\..\Data\haarcascade_eye_tree_eyeglasses.xml");

var faces = cascade.DetectMultiScale(
    image: grayImage,
    scaleFactor: 1.1,
    minNeighbors: 2,
    flags: HaarDetectionType.Zero | HaarDetectionType.ScaleImage,
    minSize: new Size(30, 30)
    );

Console.WriteLine("Detected faces: {0}", faces.Length);
```

پس از بارگذاری فایل‌های XML اطلاعات نحوه‌ی تشخیص چهره و اعضای آن، با فراخوانی متد DetectMultiScale، کار تشخیص چهره و استخراج آن از grayImage انجام خواهد شد. در اینجا minSize، اندازه‌ی حداقل چهره‌ی مدنظر است که قرار هست تشخیص داده شود. نواحی کوچکتر از این اندازه، به عنوان نویز در نظر گرفته خواهند شد و پردازش نمی‌شوند.
خروجی این متد، مستطیل‌ها و نواحی یافت شده‌ی مرتبط با چهره‌های موجود در تصویر هستند. اکنون می‌توان حلقه‌ای را تشکیل داد و این نواحی را برای مثال با مستطیل‌های رنگی، متمایز کرد:

```
var rnd = new Random();
var count = 1;
foreach (var faceRect in faces)
{
    var detectedFaceImage = new Mat(srcImage, faceRect);
    Cv2.ImShow(string.Format("Face {0}", count), detectedFaceImage);
    Cv2.WaitKey(1); // do events

    var color = Scalar.FromRgb(rnd.Next(0, 255), rnd.Next(0, 255), rnd.Next(0, 255));
    Cv2.Rectangle(srcImage, faceRect, color, 3);

    count++;
}

Cv2.ImShow("Haar Detection", srcImage);
Cv2.WaitKey(1); // do events
```

در اینجا علاوه بر رسم یک مستطیل، به دور ناحیه‌ی تشخیص داده شده، نحوه‌ی استخراج تصویر آن ناحیه را هم در سطر var detectedFaceImage مشاهده می‌کنید.

همچنین اگر علاقمند باشیم تا در این ناحیه‌ی یافت شده، چشمان شخص را نیز استخراج کنیم، می‌توان به نحو ذیل عمل کرد:

```
var rnd = new Random();
var count = 1;
foreach (var faceRect in faces)
{
    var detectedFaceImage = new Mat(srcImage, faceRect);
    Cv2.ImShow(string.Format("Face {0}", count), detectedFaceImage);
    Cv2.WaitKey(1); // do events

    var color = Scalar.FromRgb(rnd.Next(0, 255), rnd.Next(0, 255), rnd.Next(0, 255));
    Cv2.Rectangle(srcImage, faceRect, color, 3);


    var detectedFaceGrayImage = new Mat();
    Cv2.CvtColor(detectedFaceImage, detectedFaceGrayImage, ColorConversion.BgrToGray);
    var nestedObjects = nestedCascade.DetectMultiScale(
        image: detectedFaceGrayImage,
        scaleFactor: 1.1,
        minNeighbors: 2,
        flags: HaarDetectionType.Zero | HaarDetectionType.ScaleImage,
        minSize: new Size(30, 30)
        );
```

```
    Console.WriteLine("Nested Objects[{0}]: {1}", count, nestedObjects.Length);

    foreach (var nestedObject in nestedObjects)
    {
        var center = new Point
        {
            X = Cv.Round(nestedObject.X + nestedObject.Width * 0.5) + faceRect.Left,
            Y = Cv.Round(nestedObject.Y + nestedObject.Height * 0.5) + faceRect.Top
        };
        var radius = Cv.Round((nestedObject.Width + nestedObject.Height) * 0.25);
        Cv2.Circle(srcImage, center, radius, color, thickness: 3);
    }

    count++;
}

Cv2.ImShow("Haar Detection", srcImage);
Cv2.WaitKey(1); // do events
```

کدهای ابتدایی آن همانند توضیحات قبل است. تنها تفاوت آن، استفاده از nestedCascade بارگذاری شده‌ی در ابتدای بحث می‌باشد. این nestedCascade حاوی trained data استخراج چشمان اشخاص، از تصاویر است. پارامتر ورودی آن‌را نیز تصویر سیاه و سفید ناحیه‌ی چهره‌ی یافت شده، قرار داده‌ایم تا سرعت تشخیص چشمان شخص، افزایش یابد.

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