عنوان: OpenCVSharp #10
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۲۱ ۱۶:۴۵
آدرس: www.dotnettips.info
گروه‌ها: OpenCV

محاسبه و ترسیم Histogram تصاویر

هیستوگرام یک تصویر ، توزیع میزان روشنایی آن تصویر را نمایش می‌دهد و در آن تعداد نقاط قسمت‌های روشن تصویر، ترسیم می‌شوند. محاسبه‌ی هیستوگرام تصاویر در حین دیباگ الگوریتم‌های پردازش تصویر، کاربرد زیادی دارند.
OpenCV به همراه متد توکاری است به نام cv::calcHist که قادر است هیستوگرام تعدادی آرایه را محاسبه کند و در C++ API آن قرار دارد. البته هدف اصلی این متد، انجام محاسبات مرتبط است و در اینجا قصد داریم این محاسبات را نمایش دهیم.

تغییر میزان روشنایی و وضوح تصاویر در OpenCV

همانطور که عنوان شد، کار هیستوگرام تصاویر، نمایش توزیع میزان روشنایی نقاط و اجزای آن‌ها است. بنابراین می‌توان جهت مشاهده‌ی تغییر هیستوگرام محاسبه شده با تغییر میزان روشنایی و وضوح تصویر، از متد ذیل کمک گرفت:

```
private static void updateBrightnessContrast(Mat src, Mat modifiedSrc, int brightness, int contrast)
{
    brightness = brightness - 100;
    contrast = contrast - 100;

    double alpha, beta;
    if (contrast > 0)
    {
        double delta = 127f * contrast / 100f;
        alpha = 255f / (255f - delta * 2);
        beta = alpha * (brightness - delta);
    }
    else
    {
        double delta = -128f * contrast / 100;
        alpha = (256f - delta * 2) / 255f;
        beta = alpha * brightness + delta;
    }
    src.ConvertTo(modifiedSrc, MatType.CV_8UC3, alpha, beta);
}
```

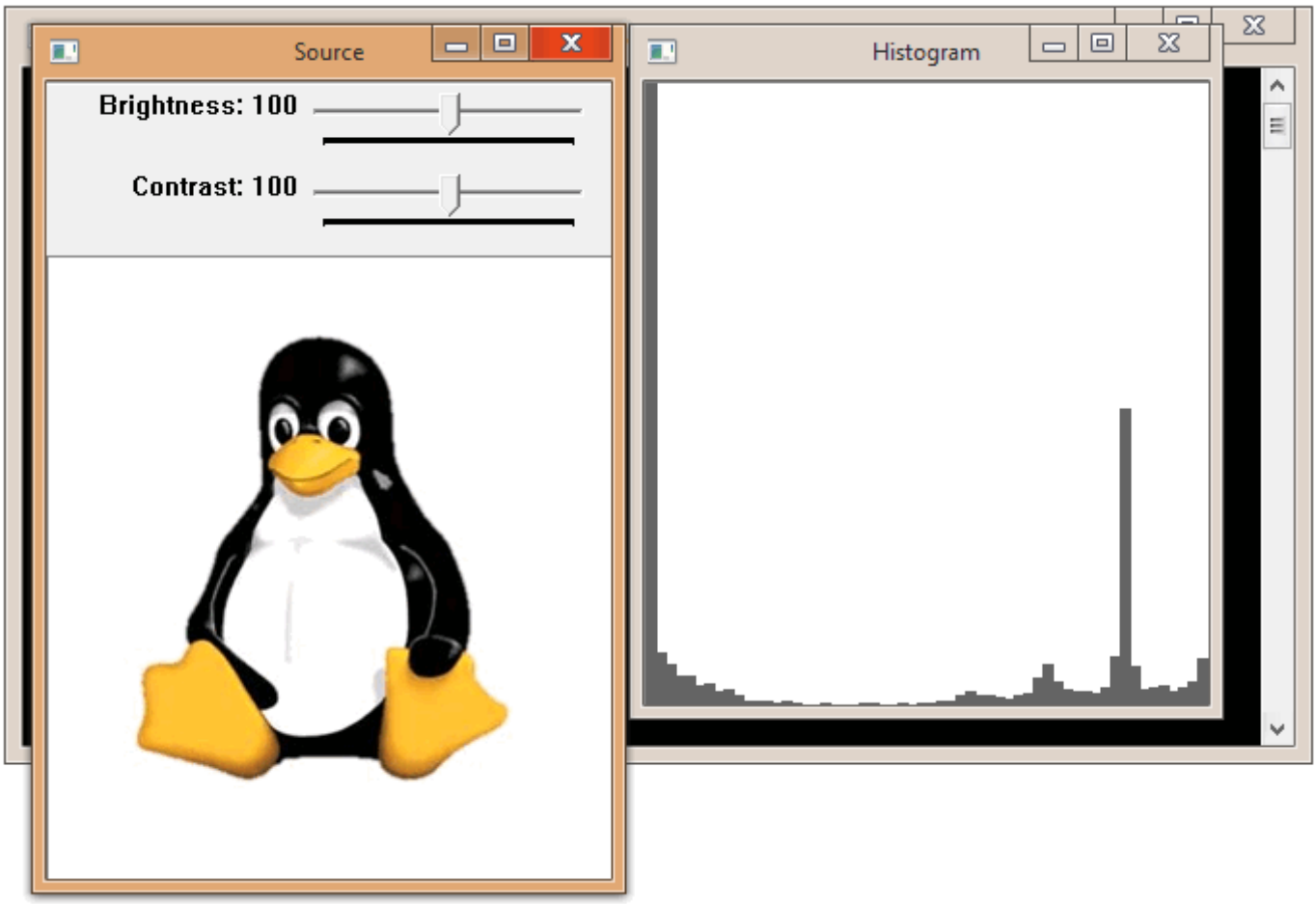
در اینجا src تصویر اصلی است. brightness و contrast، مقادیر میزان روشنایی و وضوح دریافتی از کاربر هستند. این مقادیر را می‌توان به متد ConvertTo ارسال کرد تا src را تبدیل به modifiedSrc نماید و وضوح و روشنایی آن‌را تغییر دهد.

پس از اینکه متد تغییر وضوح تصویر اصلی را تهیه کردیم، می‌توان به پنجره‌ی نمایش تصویر اصلی، دو tracker جهت دریافت brightness و contrast اضافه کرد و به این ترتیب امکان نمایش پویای تغییرات را مهیا نمود:

```
using (var src = new Mat(@"..\..\Images\Penguin.Png", LoadMode.AnyDepth | LoadMode.AnyColor))
{
    using (var sourceWindow = new Window("Source", image: src,
           flags: WindowMode.AutoSize | WindowMode.FreeRatio))
    {
        using (var histogramWindow = new Window("Histogram",
               flags: WindowMode.AutoSize | WindowMode.FreeRatio))
        {
            var brightness = 100;
            var contrast = 100;

            var brightnessTrackbar = sourceWindow.CreateTrackbar(
                    name: "Brightness", value: brightness, max: 200,
                    callback: pos =>
                    {
                        brightness = pos;
                        updateImageCalculateHistogram(sourceWindow, histogramWindow, src, brightness,
contrast);
                    });
```

```
            var contrastTrackbar = sourceWindow.CreateTrackbar(
                name: "Contrast", value: contrast, max: 200,
                callback: pos =>
                {
                    contrast = pos;
                    updateImageCalculateHistogram(sourceWindow, histogramWindow, src, brightness,
contrast);
                });


            brightnessTrackbar.Callback.DynamicInvoke(brightness);
            contrastTrackbar.Callback.DynamicInvoke(contrast);

            Cv2.WaitKey();
        }
    }
}
```

در اینجا src تصویر اصلی است. پنجره‌ی Source کار نمایش تصویر اصلی را به عهده دارد. همچنین به این پنجره، دو tracker اضافه شده‌اند تا کار دریافت مقادیر روشنایی و وضوح را از کاربر، مدیریت کنند.
پنجره‌ی دومی نیز به نام هیستوگرام در اینجا تعریف شده‌است. در این پنجره قصد داریم هیستوگرام تغییرات پویای تصویر اصلی را نمایش دهیم.



روش محاسبه‌ی هیستوگرام تصاویر و نمایش آن‌ها در OpenCVSharp

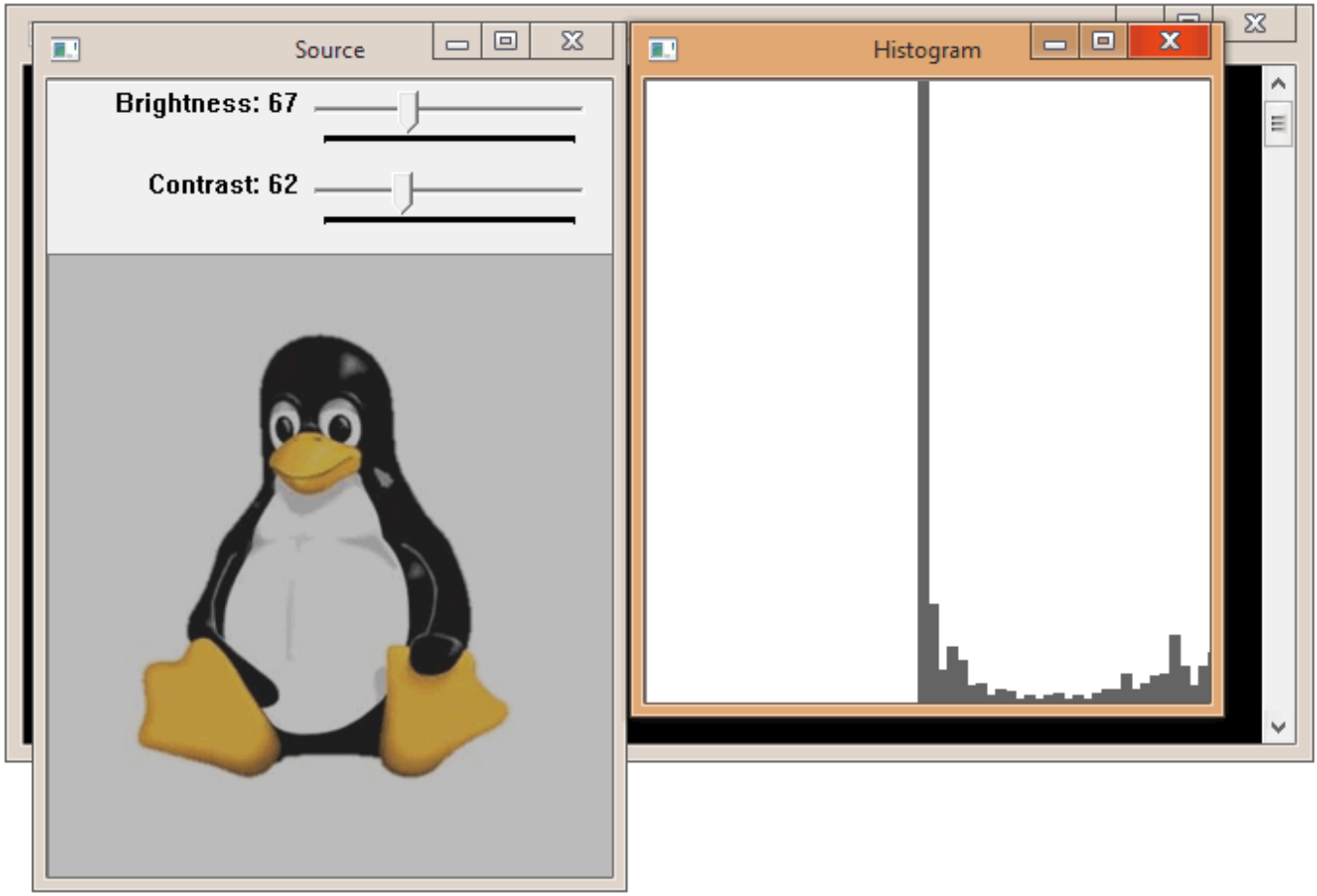
کدهای کامل محاسبه‌ی هیستوگرام تصویر اصلی تغییر یافته (modifiedSrc) و سپس نمایش آن‌را در پنجره‌ی histogramWindow،

در ادامه ملاحظه می‌کنید:

```
private static void calculateHistogram1(Window histogramWindow, Mat src, Mat modifiedSrc)
{
    const int histogramSize = 64;
    int[] dimensions = { histogramSize }; // Histogram size for each dimension
    Rangef[] ranges = { new Rangef(0, histogramSize) }; // min/max

    using (var histogram = new Mat())
    {
        Cv2.CalcHist(
            images: new[] { modifiedSrc },
            channels: new[] { 0 },
            mask: null,
            hist: histogram,
            dims: 1,
            histSize: dimensions,
            ranges: ranges);

        using (var histogramImage = (Mat)(Mat.Ones(rows: src.Rows, cols: src.Cols, type: MatType.CV_8U)
* 255))
        {
            // Scales and draws histogram

            Cv2.Normalize(histogram, histogram, 0, histogramImage.Rows, NormType.MinMax);
            var binW = Cv.Round((double)histogramImage.Cols / histogramSize);

            var color = Scalar.All(100);

            for (var i = 0; i < histogramSize; i++)
            {
                Cv2.Rectangle(histogramImage,
                    new Point(i * binW, histogramImage.Rows),
                    new Point((i + 1) * binW, histogramImage.Rows - Cv.Round(histogram.Get<float>(i))),
                    color,
                    -1);
            }

            histogramWindow.Image = histogramImage;
        }
    }
}
```

معادل متد cv::calcHist، متد Cv2.CalcHist در OpenCVSharp است. این متد آرایه‌ای از تصاویر را قبول می‌کند که در اینجا تنها قصد داریم با یک تصویر کار کنیم. به همین جهت آرایه‌های images، اندازه‌های آن‌ها و بازه‌های min/max این تصاویر تنها یک عضو دارند. خروجی این متد پارامتر hist آن است که توسط یک new Mat تامین شده‌است. مقدار dims به یک تنظیم شده‌است؛ زیرا در اینجا تنها قصد داریم شدت نقاط را اندازه گیری کنیم. پارامتر ranges مشخص می‌کند که مقادیر اندازه گیری شده باید در چه بازه‌ایی جمع آوری شوند.

پس از محاسبه‌ی هیستوگرام، یک تصویر خالی پر شده‌ی با عدد یک را توسط متد Mat.Ones ایجاد می‌کنیم. این تصویر به عنوان منبع تصویر هیستوگرام نمایش داده شده، مورد استفاده قرار می‌گیرد. سپس نیاز است اطلاعات محاسبه شده، در مقیاسی قرار گیرند که قابل نمایش باشد. به همین جهت با استفاده از متد Normalize، آن‌ها را در مقیاس و بازه‌ی ارتفاع تصویر، تغییر اندازه خواهیم داد. سپس به کمک متد مستطیل، خروجی آرایه هیستوگرام را در صفحه، با رنگ خاکستری مشخص شده توسط متد Scalar.All ترسیم خواهیم کرد.



همانطور که در این تصویر ملاحظه می‌کنید، با کدرتر شدن تصویر اصلی، هیستوگرام آن، توزیع روشنایی کمتری را نمایش می‌دهد.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید.