عنوان: **OpenCVSharp #7**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۱۷ ۱۳:۱۵
آدرس: www.dotnettips.info
گروه‌ها: OpenCV

معرفی اینترفیس C++ کتابخانه‌ی OpenCVSharp

اینترفیس یا API زبان C کتابخانه‌ی OpenCV مربوط است به نگارش‌های 1x این کتابخانه و تمام مثال‌هایی را که تاکنون ملاحظه کردید، بر مبنای همین اینترفیس تهیه شده بودند. اما از OpenCV سری 2x، این اینترفیس صرفا جهت سازگاری با نگارش‌های قبلی، نگهداری می‌شود و اینترفیس اصلی مورد استفاده، API جدید C++ آن است. به همین جهت کتابخانه‌ی OpenCVSharp نیز در فضای نام OpenCvSharp.CPlusPlus و توسط اسمبلی OpenCvSharp.CPlusPlus.dll، امکان دسترسی به این API جدید را فراهم کرده‌است که در ادامه نکات مهم آن‌را بررسی خواهیم کرد.

تبدیل مثال‌های اینترفیس C به اینترفیس C++

مثال « تبدیل تصویر به حالت سیاه و سفید » قسمت سوم را درنظر بگیرید. این مثال به کمک اینترفیس C کتابخانه‌ی OpenCV کار می‌کند. معادل تبدیل شده‌ی آن به اینترفیس C++ به صورت ذیل است:

```
// Cv2.ImRead
using (var src = new Mat(@"..\..\Images\Penguin.Png", LoadMode.AnyDepth | LoadMode.AnyColor))
using (var dst = new Mat())
{
    Cv2.CvtColor(src, dst, ColorConversion.BgrToGray);

    // How to export
    using (var bitmap = dst.ToBitmap()) // => OpenCvSharp.Extensions.BitmapConverter.ToBitmap(dst)
    {
        bitmap.Save("gray.png", ImageFormat.Png);
    }

    using (new Window("BgrToGray C++: src", image: src))
    using (new Window("BgrToGray C++: dst", image: dst))
    {
        Cv2.WaitKey();
    }
}
```

نکاتی را که باید در اینجا مدنظر داشت:
- بجای IplImage، از کلاس Mat استفاده شده‌است.
- برای ایجاد Clone یک تصویر نیازی نیست تا پارامترهای خاصی را به Mat دوم (همان dst) انتساب داد و ایجاد یک Mat خالی کفایت می‌کند.
- اینبار بجای کلاس Cv اینترفیس C، از کلاس Cv2 اینترفیس C++ استفاده شده‌است.
- متد الحاقی ToBitmap نیز که در کلاس OpenCvSharp.Extensions.BitmapConverter قرار دارد، با نمونه‌ی Mat سازگار است و به این ترتیب می‌توان خروجی معادل دات نتی Mat را با فرمت Bitmap تهیه کرد.
- بجای CvWindow، در اینجا باید از Window سازگار با Mat، استفاده شود.
- new Mat معادل Cv2.ImRead است. بنابراین اگر مثال C++ ایی را در اینترنت یافتید:

```
cv::Mat src = cv::imread ("foo.jpg");
cv::Mat dst;
cv::cvtColor (src, dst, CV_BGR2GRAY);
```

معادل متد imread آن همان new Mat کتابخانه‌ی OpenCVSharp است و یا متد Cv2.ImRead آن.

کار مستقیم با نقاط در OpenCVSharp

متدهای ماتریسی OpenCV، فوق العاده در جهت سریع اجرا شدن و استفاده‌ی از امکانات سخت افزاری و پردازش‌های موازی، بهینه سازی شده‌اند. اما اگر قصد داشتید این متدهای سریع را با نمونه‌هایی متداول و نه چندان سریع جایگزین کنید، می‌توان مستقیما با نقاط تصویر نیز کار کرد. در ادامه قصد داریم کار فیلتر توکار Not را که عملیات معکوس سازی رنگ نقاط را انجام می‌دهد، شبیه سازی کنیم.

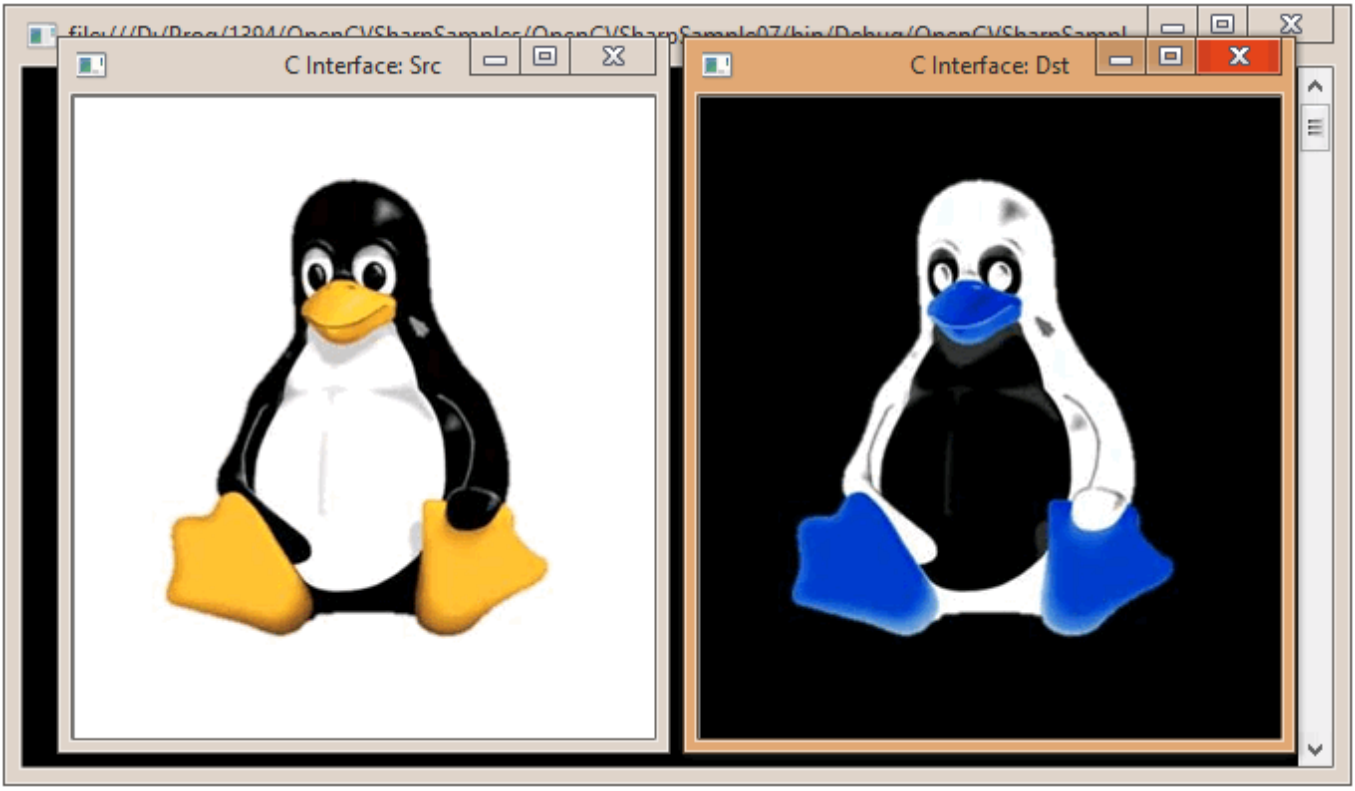
در اینجا نحوه‌ی دسترسی مستقیم به نقاط تصویر بارگذاری شده را توسط اینترفیس C، ملاحظه می‌کنید:

```
using (var src = new IplImage(@"..\..\Images\Penguin.Png", LoadMode.AnyDepth | LoadMode.AnyColor))
using (var dst = new IplImage(src.Size, src.Depth, src.NChannels))
{
    for (var y = 0; y < src.Height; y++)
    {
        for (var x = 0; x < src.Width; x++)
        {
            CvColor pixel = src[y, x];
            dst[y, x] = new CvColor
            {
                B = (byte)(255 - pixel.B),
                G = (byte)(255 - pixel.G),
                R = (byte)(255 - pixel.R)
            };
        }
    }

    // [C] Accessing Pixel
    // https://github.com/shimat/opencvsharp/wiki/%5BC%5D-Accessing-Pixel

    using (new CvWindow("C Interface: Src", image: src))
    using (new CvWindow("C Interface: Dst", image: dst))
    {
        Cv.WaitKey(0);
    }
}
```

IplImage امکان دسترسی به نقاط را به صورت یک آرایه‌ی دو بعدی میسر می‌کند. خروجی آن از نوع CvColor است که در اینجا از هر عنصر آن، 255 واحد کسر خواهد شد تا فیلتر Not شبیه سازی شود. سپس این رنگ جدید، به نقطه‌ای معادل آن در تصویر خروجی انتساب داده می‌شود.
روش ارائه شده‌ی در اینجا یکی از روش‌های دسترسی به نقاط، توسط اینترفیس C است. سایر روش‌های ممکن را در Wiki آن می‌توانید مطالعه کنید.



شبیه به همین کار را می‌توان به نحو ذیل توسط اینترفیس C++ کتابخانه‌ی OpenCVSharp نیز انجام داد:

```
// Cv2.ImRead
using (var src = new Mat(@"..\..\Images\Penguin.Png", LoadMode.AnyDepth | LoadMode.AnyColor))
using (var dst = new Mat())
{
    src.CopyTo(dst);

    for (var y = 0; y < src.Height; y++)
    {
        for (var x = 0; x < src.Width; x++)
        {
            var pixel = src.Get<Vec3b>(y, x);
            var newPixel = new Vec3b
            {
                Item0 = (byte)(255 - pixel.Item0), // B
                Item1 = (byte)(255 - pixel.Item1), // G
                Item2 = (byte)(255 - pixel.Item2) // R
            };
            dst.Set(y, x, newPixel);
        }
    }

    // [Cpp] Accessing Pixel
    // https://github.com/shimat/opencvsharp/wiki/%5BCpp%5D-Accessing-Pixel

    //Cv2.NamedWindow();
    //Cv2.ImShow();
    using (new Window("C++ Interface: Src", image: src))
    using (new Window("C++ Interface: Dst", image: dst))
    {
        Cv2.WaitKey(0);
    }
}
```

ابتدا توسط کلاس Mat، کار بارگذاری و سپس تهیه‌ی یک کپی، از تصویر اصلی انجام می‌شود. در ادامه برای دسترسی به نقاط تصویر، از متد Get که خروجی آن از نوع Vec3b است، استفاده خواهد شد. این بردار دارای سه جزء است که بیانگر اجزای رنگ نقطه‌ی مدنظر می‌باشند. در اینجا نیز 255 واحد از هر جزء کسر شده و سپس توسط متد Set، به تصویر خروجی اعمال خواهند

شد.
می‌توانید سایر روش‌های دسترسی به نقاط را توسط اینترفیس C++، در Wiki این کتابخانه مطالعه نمائید.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید.