عنوان: **OpenCVSharp #4**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۱۴ ۱۳:۰
آدرس: www.dotnettips.info
گروه‌ها: OpenCV

کار با فیلترها در OpenCVSharp

فرض کنید قصد داریم یک چنین مثال زبان C را که در مورد کار با فیلترها در OpenCV است، به نمونه‌ی دات نتی آن تبدیل کنیم:

```
#include <cv.h>
        #include <highgui.h>
        #include <stdio.h>

        int main (int argc, char **argv)
        {
          IplImage *src_img = 0, *dst_img;
          float data[] = { 2, 2, 2, 2, 2, 2, 2, 2, 2,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1
          };
          CvMat kernel = cvMat (1, 21, CV_32F, data);

          if (argc >= 2)
            src_img = cvLoadImage (argv[1], CV_LOAD_IMAGE_ANYDEPTH | CV_LOAD_IMAGE_ANYCOLOR);
          if (src_img == 0)
            exit (-1);

          dst_img = cvCreateImage (cvGetSize (src_img), src_img->depth, src_img->nChannels);

          cvNormalize (&kernel, &kernel, 1.0, 0, CV_L1);
          cvFilter2D (src_img, dst_img, &kernel, cvPoint (0, 0));

          cvNamedWindow ("Filter2D", CV_WINDOW_AUTOSIZE);
          cvShowImage ("Filter2D", dst_img);
          cvWaitKey (0);

          cvDestroyWindow ("Filter2D");
          cvReleaseImage (&src_img);
          cvReleaseImage (&dst_img);

          return 0;
        }
```

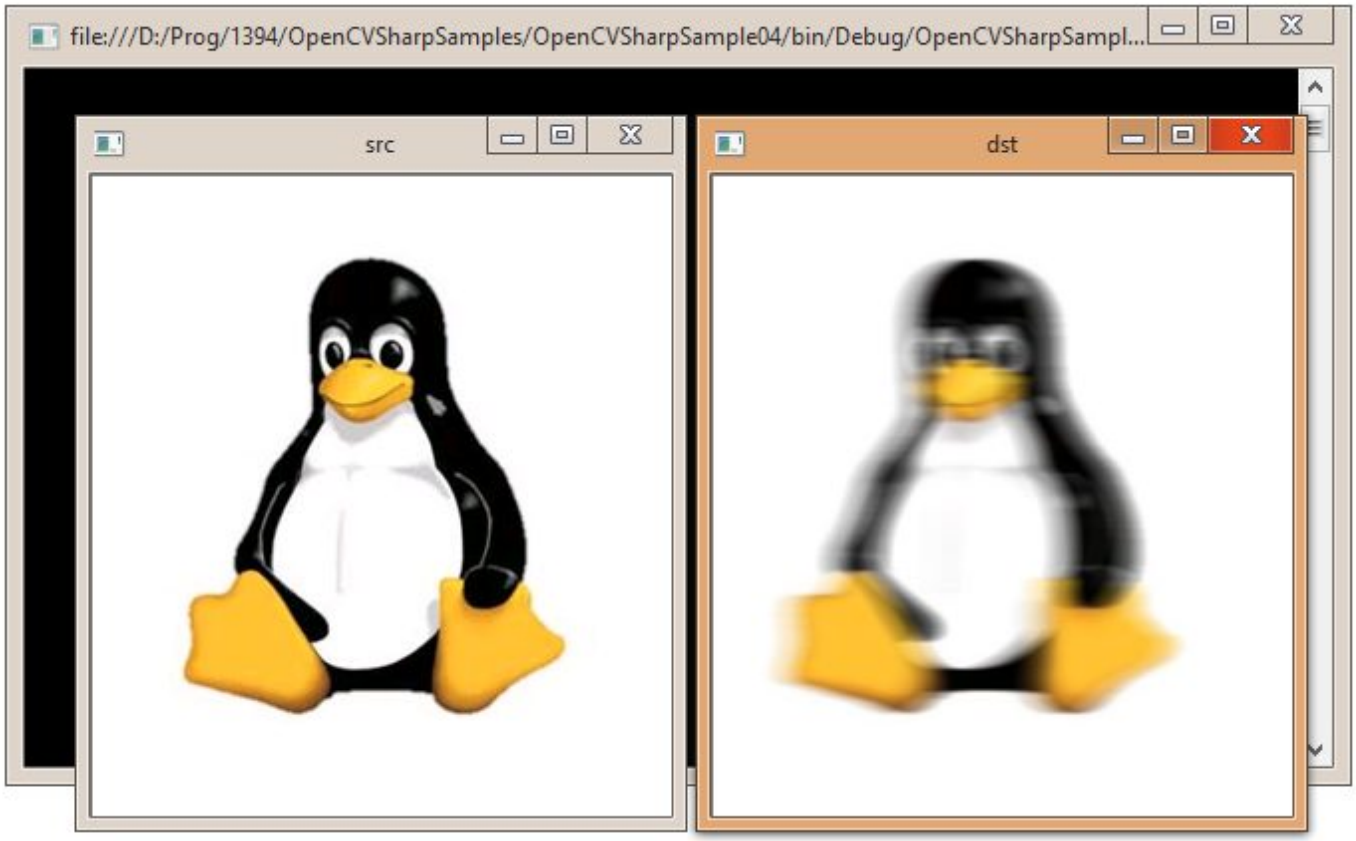
معادل OpenCVSharp آن به صورت ذیل خواهد بود:

```
using (var src = new IplImage(@"..\..\Images\Penguin.Png", LoadMode.AnyDepth | LoadMode.AnyColor))
using (var dst = new IplImage(src.Size, src.Depth, src.NChannels))
{
    float[] data = { 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
    var kernel = new CvMat(rows: 1, cols: 21, type: MatrixType.F32C1, elements: data);

    Cv.Normalize(src: kernel, dst: kernel, a: 1.0, b: 0, normType: NormType.L1);
    Cv.Filter2D(src, dst, kernel, anchor: new CvPoint(0, 0));

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

با این خروجی:



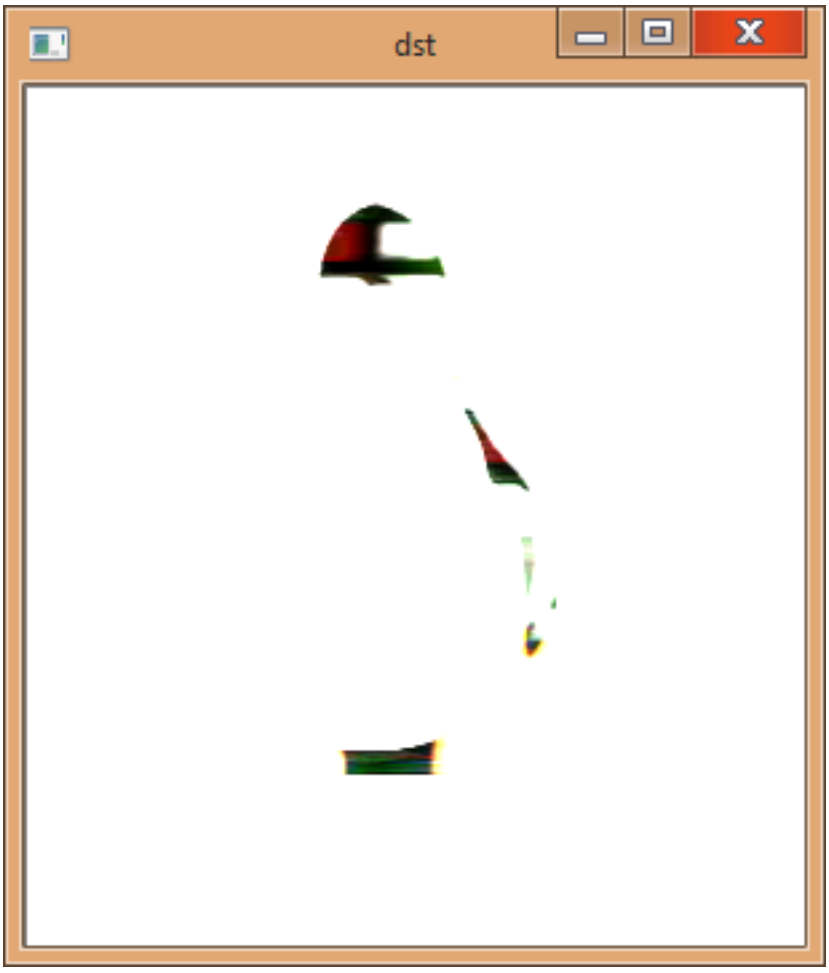
در قسمت‌های قبلی در مورد بارگذاری تصاویر، تهیه‌ی یک Clone از آن و همچنین ساخت یک پنجره به روش‌های مختلف و رها سازی خودکار منابع مرتبط، بیشتر بحث شد. در اینجا تصویر اصلی با همان عمق و وضوح تغییر نیافته‌ی آن بارگذاری می‌شود.
کار cvMat ، آغاز یک ماتریس OpenCV است. پارامترهای آن، تعداد ردیف‌ها، ستون‌ها، نوع داده‌ی المان‌ها و داده‌های مرتبط را مشخص می‌کنند.
در این مثال آرایه‌ی data، یک فیلتر را تعریف می‌کند که در اینجا، حالت یک بردار را دارد تا یک ماتریس. برای تبدیل آن به ماتریس، از شیء CvMat استفاده خواهد شد که آن‌را تبدیل به ماتریسی با یک ردیف و 21 ستون خواهد کرد.
در اینجا از نام کرنل استفاده شده‌است. کرنل در OpenCV به معنای ماتریسی از داده‌ها با یک نقطه‌ی anchor (لنگر) است. این لنگر به صورت پیش فرض در میانه‌ی ماتریس قرار دارد (نقطه‌ی 1- , 1-).

1	-2	1
2	-4	2
1	-2	1

مرحله‌ی بعد، نرمال سازی این فیلتر است. تاثیر نرمال سازی اطلاعات را به این نحو می‌توان نمایش داد:

```
double sum = 0;
foreach (var item in data)
{
     sum += Math.Abs(item);
}
Console.WriteLine(sum); // => .999999970197678
```

در اینجا پس از نرمال سازی، جمع عناصر بردار data، تقریبا مساوی 1 خواهد بود و تمام عناصر بردار data، به داده‌هایی بین یک و صفر، نگاشت خواهند شد.
اگر مرحله‌ی نرمال سازی اطلاعات را حذف کنیم، تصویر نهایی حاصل، چنین شکلی را پیدا می‌کند:



زیرا عملیات تغییر اندازه‌ی اطلاعات بردار صورت نگرفته‌است و داده‌های آن مطلوب متد cvFilter2D نیست.
و مرحله‌ی آخر، اجرای این بردار نرمال شده خطی، بر روی تصویر اصلی به کمک متد cvFilter2D است. این متد، تصویر مبدا را پس از تبدیلات ماتریسی، به تصویر مقصد تبدیل می‌کند. فرمول ریاضی اعمال شده‌ی در اینجا برای محاسبه‌ی نقاط تصویر خروجی به صورت زیر است:

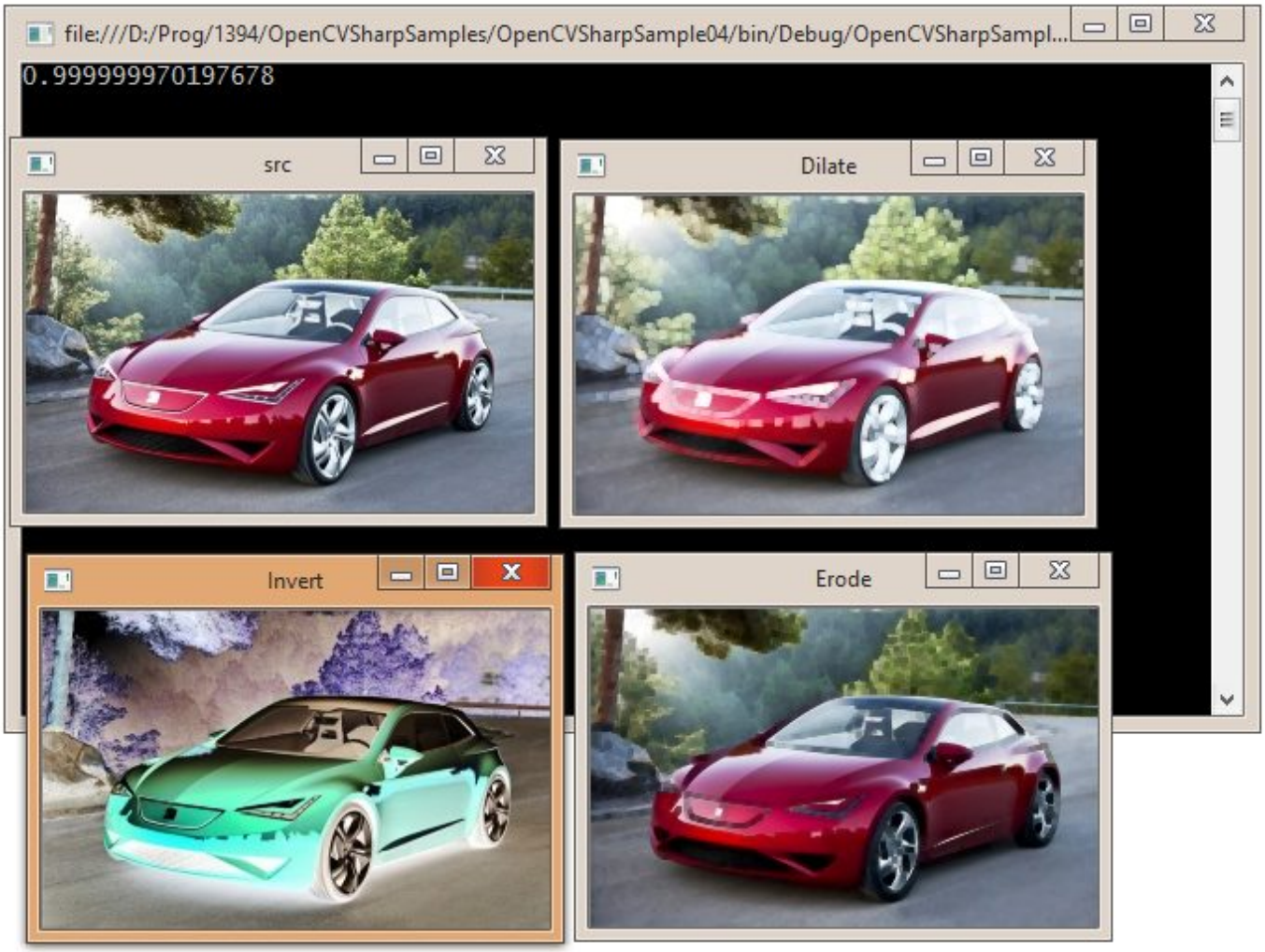
$$\mathrm{dst}(x,y) = \sum_{\substack{0 \le x' < \mathrm{kernel.cols}, \\ 0 \le y' < \mathrm{kernel.rows}}} \mathrm{kernel}(x',y') * \mathrm{src}(x + x' - \mathrm{anchor.x}, y + y' - \mathrm{anchor.y})$$

فیلترهای توکار OpenCV

علاوه بر امکان طراحی فیلترهای سفارشی خطی مانند مثال فوق، کتابخانه‌ی OpenCV دارای تعدادی فیلتر توکار نیز می‌باشد که نمونه‌ای از آن‌را در مثال ذیل می‌توانید مشاهده کنید:

```
using (var src = new IplImage(@"..\..\Images\Car.jpg", LoadMode.AnyDepth | LoadMode.AnyColor))
{
    using (var dst = new IplImage(src.Size, src.Depth, src.NChannels))
    {
        using (new CvWindow("src", image: src))
        {
            Cv.Erode(src, dst);
            using (new CvWindow("Erode", image: dst))
            {
                Cv.Dilate(src, dst);
                using (new CvWindow("Dilate", image: dst))
                {
                    Cv.Not(src, dst);
                    using (new CvWindow("Invert", image: dst))
                    {
                        Cv.WaitKey(0);
                    }
                }
            }
        }
    }
}
```

با این خروجی



کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید.

نظرات خوانندگان

نویسنده: محسن نجف زاده
تاریخ: ۱۳۹۴/۰۴/۰۴ ۱۰:۳۶

سلام / چرا کرنل ی که برای بلور کردن تصویر مثال اول استفاده شده 1 در 21 است؟ (و آیا لزوم نباید یک کرنل مربعی استفاده شه)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۰۴ ۱۱:۵۷

اندازه‌ی کرنل برای کار با بسیاری از متدهای دیگر باید یک عدد فرد باشد (در مورد عدد 21) و ... مربعی هست. در حقیقت یک ماتریس یک سطری با 21 ستون است (شیء CvMat تعریف شده از آن).