عنوان: OpenCVSharp #6
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۱۶ ۱۰:۴۵
آدرس: www.dotnettips.info
گروه‌ها: OpenCV

نمایش ویدیو و اعمال فیلتر بر روی آن

در قسمت قبل با نحوه‌ی نمایش تصاویر OpenCV در برنامه‌های دات نتی آشنا شدیم. در این قسمت قصد داریم همان نکات را جهت پخش یک ویدیو توسط OpenCVSharp بسط دهیم.

روش‌های متفاوت پخش ویدیو و یا کار با یک Capture Device

OpenCV امکان کار با یک WebCam، دوربین و یا فیلم‌های آماده را دارد. برای این منظور کلاس CvCapture در OpenCVSharp پیش بینی شده‌است. در اینجا قصد داریم جهت سهولت پیگیری بحث، یک فایل avi را به عنوان منبع CvCapture معرفی کنیم:

```
using (var capture = new CvCapture(@"..\..\Videos\drop.avi"))
{
    var image = capture.QueryFrame();
}
```

روش کلی کار با CvCapture را در اینجا ملاحظه می‌کنید. متد QueryFrame هربار یک frame از ویدیو را بازگشت می‌دهد و می‌توان آن‌را در یک حلقه، تا زمانیکه image نال بازگشت داده نشده، ادامه داد. همچنین برای نمایش آن نیز می‌توان از یکی از روش‌های مطرح شده ، مانند picture box استاندارد یا PictureBoxIpl (روش توصیه شده) استفاده کرد. اگر از PictureBoxIpl استفاده می‌کنید، متد pictureBoxIpl1.RefreshIplImage آن دقیقا برای یک چنین مواردی طراحی شده‌است تا سربار نمایش تصاویر را به حداقل برساند.
در اینجا اولین روشی که جهت به روز رسانی UI به نظر می‌رسد، استفاده از متد Application.DoEvents است تا UI فرصت داشته باشد، تعداد فریم‌های بالا را نمایش دهد و خود را به روز کند:

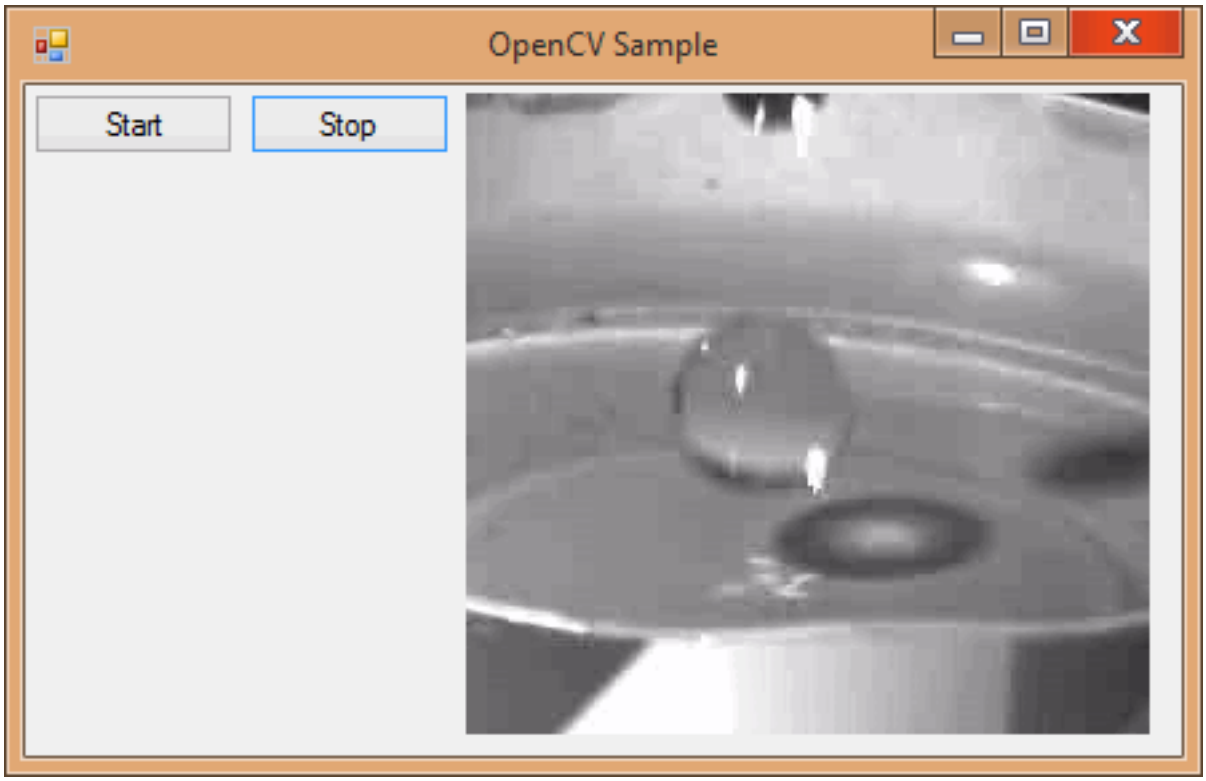
```
IplImage image;
while ((image = Capture.QueryFrame()) != null)
{
    _pictureBoxIpl1.RefreshIplImage(image);

    Thread.Sleep(interval);
    Application.DoEvents();
}
```

این روش هرچند کار می‌کند اما همانند روش استفاده از متد رخدادگردان Application Do Idle که صرفا در زمان بیکاری برنامه فراخوانی می‌شود، سبب خواهد شد تا تعدادی فریم را از دست دهید، همچنین با CPU Usage بالایی نیز مواجه شوید.
روش بعدی، استفاده از یک تایمر است که Interval آن بر اساس نرخ فریم‌های ویدیو تنظیم شده‌است:

```
timer = new Timer();
timer.Interval = (int)(1000 / Capture.Fps);
timer.Tick += Timer_Tick;
```

این روش بهتر است از روش DoEvents و به خوبی کار می‌کند؛ اما باز هم کار دریافت و همچنین پخش فریم‌ها، در ترد اصلی برنامه انجام خواهد شد.
روش بهتر از این، انتقال دریافت فریم‌ها به تردی جداگانه و پخش آن‌ها در ترد اصلی برنامه است؛ زیرا نمی‌توان GUI را از طریق یک ترد دیگر به روز رسانی کرد. برای این منظور می‌توان از BackgroundWorker دات نت کمک گرفت. رخداد DoWork آن در تردی جداگانه و مجزای از ترد اصلی برنامه اجرا می‌شود، اما رخداد ProgressChanged آن در ترد اصلی برنامه اجرا شده و امکان به روز رسانی UI را فراهم می‌کند.

استفاده از BackgroundWorker جهت پخش ویدیو به کمک OpenCVSharp



ابتدا دو دکمه‌ی Start و Stop را به فرم اضافه خواهیم کرد (شکل فوق).
سپس در زمان آغاز برنامه، یک PictureBoxIpl را به فرم جاری اضافه می‌کنیم:

```
private void FrmMain_Load(object sender, System.EventArgs e)
{
    _pictureBoxIpl1 = new PictureBoxIpl
    {
        AutoSize = true
    };
    flowLayoutPanel1.Controls.Add(_pictureBoxIpl1);
}
```

و یا همانطور که در قسمت پیشین نیز عنوان شد، می‌توانید این کنترل را به نوار ابزار VS.NET اضافه کرده و سپس به سادگی آن‌را روی فرم قرار دهید.

در دکمه‌ی Start، کار آغاز BackgroundWorker انجام خواهد شد:

```
private void BtnStart_Click(object sender, System.EventArgs e)
{
    if (_worker != null && _worker.IsBusy)
    {
        return;
    }

    _worker = new BackgroundWorker
    {
        WorkerReportsProgress = true,
        WorkerSupportsCancellation = true
    };
    _worker.DoWork += workerDoWork;
    _worker.ProgressChanged += workerProgressChanged;
    _worker.RunWorkerCompleted += workerRunWorkerCompleted;
```

```
    _worker.RunWorkerAsync();

    BtnStart.Enabled = false;
}
```

در اینجا یک سری خاصیت را مانند امکان لغو عملیات، جهت استفاده‌ی در دکمه‌ی Stop، به همراه تنظیم رخدادگردان‌هایی جهت دریافت و نمایش فریم‌ها تعریف کرده‌ایم. کدهای این روال‌های رخدادگردان را در ادامه ملاحظه می‌کنید:

```
private void workerDoWork(object sender, DoWorkEventArgs e)
{
    using (var capture = new CvCapture(@"..\..\Videos\drop.avi"))
    {
        var interval = (int)(1000 / capture.Fps);

        IplImage image;
        while ((image = capture.QueryFrame()) != null &&
                _worker != null && !_worker.CancellationPending)
        {
            _worker.ReportProgress(0, image);
            Thread.Sleep(interval);
        }
    }
}

private void workerProgressChanged(object sender, ProgressChangedEventArgs e)
{
    var image = e.UserState as IplImage;
    if (image == null) return;

    Cv.Not(image, image);
    _pictureBoxIpl1.RefreshIplImage(image);
}

private void workerRunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
{
    _worker.Dispose();
    _worker = null;
    BtnStart.Enabled = true;
}
```

متد workerDoWork کار دریافت فریم‌ها را در یک ترد مجزای از ترد اصلی برنامه به عهده دارد. این فریم‌ها توسط متد ReportProgress به متد workerProgressChanged جهت نمایش نهایی ارسال خواهند شد. این متد در ترد اصلی برنامه اجرا می‌شود و در اینجا کار با UI، مشکلی را به همراه نخواهد داشت و برنامه کرش نمی‌کند. اگر در متد workerDoWork کار به روز رسانی UI را مستقیما انجام دهیم، چون ترد اجرایی آن، با ترد اصلی برنامه یکی نیست، برنامه بلافاصله کرش خواهد کرد.
متد workerRunWorkerCompleted در پایان کار نمایش ویدیو، به صورت خودکار فراخوانی شده و در اینجا می‌توانیم دکمه‌ی Start را مجددا فعال کنیم.
همچنین در حین نمایش ویدیو، با کلیک بر روی دکمه‌ی Stop، می‌توان درخواست لغو عملیات را صادر کرد:

```
private void BtnStop_Click(object sender, System.EventArgs e)
{
    if (_worker != null)
    {
        _worker.CancelAsync();
        _worker.Dispose();
    }
    BtnStart.Enabled = true;
}
```

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید.

نظرات خوانندگان

نویسنده: علی ساری
تاریخ: ۱۳۹۴/۰۳/۱۷ ۹:۵۴

من از کدهای زیر استفاده کردم و در نهایت این خطا را در خط Application.Run(new Form1()); گرفتم

```
An unhandled exception of type 'System.Reflection.TargetInvocationException' occurred in mscorlib.dll

Additional information: Exception has been thrown by the target of an invocation.
```

کدهایی که در برنامه نوشتم:

```
private void button1_Click(object sender, EventArgs e)
        {
            if (_worker != null && _worker.IsBusy)
            {
                return;
            }

            _worker = new BackgroundWorker
            {
                WorkerReportsProgress = true,
                WorkerSupportsCancellation = true
            };
            _worker.DoWork += workerDoWork;
            _worker.ProgressChanged += workerProgressChanged;
            _worker.RunWorkerCompleted += workerRunWorkerCompleted;
            _worker.RunWorkerAsync();
        }
```

```
private void workerDoWork(object sender, DoWorkEventArgs e)
        {
            //var interval = (int)(1000 / _capture.Fps);
            Image image;
            while ((image = _capture.QueryFrame().ToBitmap()) != null &&
                   _worker != null && !_worker.CancellationPending)
            {
                _worker.ReportProgress(0, image);
                //Thread.Sleep(interval);

                Thread.Sleep(10);
            }
        }
        private void workerProgressChanged(object sender, ProgressChangedEventArgs e)
        {
            var image = e.UserState as Image;
            if (image == null) return;

            //Cv.Not(image, image);
            //_pictureBoxIpl1.RefreshIplImage(image);
            //_pictureBoxIpl1.Image=image;

            _pictureBoxIpl1.Invoke(new EventHandler(delegate
            {
                _pictureBoxIpl1.Image = image;
            }));


        }

        private void workerRunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
        {
            _worker.Dispose();
            _worker = null;
        }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۱۷ ۱۰:۱۳

- وجود Thread Sleep با مقداری که در مطلب فوق عنوان شده، ضروری هست. از این جهت که اساسا رابط کاربری معمولی ویندوز، قابلیت پردازش تعداد عظیمی از پیام‌های رسیده را ندارد و باید در این بین به آن فرصت داد. بحث DirectShow مجموعه‌ی Direct-X متفاوت است و طراحی اختصاصی آن برای یک چنین کارهایی است. اما در اینجا نمی‌توانید UI معمولی را با سیلی از داده‌ها و پیام‌های به روز رسانی، مدفون کنید.
- استفاده از متد capture.QueryFrame().ToBitmap اشتباه هست. از این جهت که خروجی capture.QueryFrame می‌تواند نال باشد. بنابراین این تبدیل را باید در داخل حلقه انجام دهید و نه در زمانیکه قصد دارید تصویری را دریافت کنید. شرط موجود در حلقه (مانند مثال اصلی مطلب)، بررسی نال نبودن این فریم دریافتی است. بنابراین اگر نال باشد، حلقه پایان خواهد یافت.
- همانطور که در متن عنوان شد، متد workerProgressChanged در ترد اصلی یا همان ترد UI اجرا می‌شود. بنابراین فراخوانی pictureBoxIpl1.Invoke غیر ضروری است و سربار بی‌جهتی را به سیستم تحمیل می‌کند.

به صورت خلاصه در حین استفاده‌ی از BackgroundWorker:

- متد رخداد گردان DoWork بر روی ThreadPool اجرا می‌شود (ترد آن با ترد UI یکی نیست)
- متدهای رخدادگردان گزارش پیشرفت کار و اتمام کار، بر روی ترد UI اجرا می‌شوند. بنابراین امکان دسترسی به عناصر UI در این متدها، بدون مشکلی وجود دارد.