# 转 ffmpeg中的sws_scale算法性能测试

2013年09月25日 21:18:41 阅读数：17255

经常用到ffmpeg中的sws_scale来进行图像缩放和格式转换，该函数可以使用各种不同算法来对图像进行处理。以前一直很懒，懒得测试和甄别应该使用哪种算法，最近的工作时间，很多时候需要等待别人。忙里偷闲，对ffmpeg的这一组函数进行了一下封装，顺便测试了一下各种算法。

简单说一下测试环境，我使用的是Dell的品牌机，i5的CPU。ffmpeg是2010年8月左右的当时最新版本编译而成，我使用的是其静态库版本。

sws_scale的算法有如下这些选择。

```cpp
#define SWS_FAST_BILINEAR     1
#define SWS_BILINEAR          2
#define SWS_BICUBIC           4
#define SWS_X                 8
#define SWS_POINT          0x10
#define SWS_AREA           0x20
#define SWS_BICUBLIN       0x40
#define SWS_GAUSS          0x80
#define SWS_SINC          0x100
#define SWS_LANCZOS       0x200
#define SWS_SPLINE        0x400
```

首先，将一幅1920*1080的风景图像，缩放为400*300的24位RGB，下面的帧率，是指每秒钟缩放并渲染的次数。（经过我的测试，渲染的时间可以忽略不计，主要时间还是耗费在缩放算法上。）

| 算法 | 帧率 | 图像主观感受 |
| --- | --- | --- |
| SWS_FAST_BILINEAR | 228 | 图像无明显失真，感觉效果很不错。 |
| SWS_BILINEAR | 95 | 感觉也很不错，比上一个算法边缘平滑一些。 |
| SWS_BICUBIC | 80 | 感觉差不多，比上上算法边缘要平滑，比上一算法要锐利。 |
| SWS_X | 91 | 与上一图像，我看不出区别。 |
| SWS_POINT | 427 | 细节比较锐利，图像效果比上图略差一点点。 |
| SWS_AREA | 116 | 与上上算法，我看不出区别。 |
| SWS_BICUBLIN | 87 | 同上。 |
| SWS_GAUSS | 80 | 相对于上一算法，要平滑(也可以说是模糊)一些。 |
| SWS_SINC | 30 | 相对于上一算法，细节要清晰一些。 |
| SWS_LANCZOS | 70 | 相对于上一算法，要平滑(也可以说是模糊)一点点，几乎无区别。 |
| SWS_SPLINE | 47 | 和上一个算法，我看不出区别。 |

总评，以上各种算法，图片缩小之后的效果似乎都不错。如果不是对比着看，几乎看不出缩放效果的好坏。上面所说的清晰（锐利）与平滑（模糊），是一种客观感受，并非清晰就比平滑好，也非平滑比清晰好。其中的Point算法，效率之高，让我震撼，但效果却不差。此外，我对比过使用CImage的绘制时缩放，其帧率可到190，但效果惨不忍睹，颜色严重失真。

第二个试验，将一幅1024*768的风景图像，放大到1920*1080，并进行渲染（此时的渲染时间，虽然不是忽略不计，但不超过5ms的渲染时间，不影响下面结论的相对准确性）。

| 算法 | 帧率 | 图像主观感受 |
| --- | --- | --- |
| SWS_FAST_BILINEAR | 103 | 图像无明显失真，感觉效果很不错。 |
| SWS_BILINEAR | 100 | 和上图看不出区别。 |
| SWS_BICUBIC | 78 | 相对上图，感觉细节清晰一点点。 |
| SWS_X | 106 | 与上上图无区别。 |
| SWS_POINT | 112 | 边缘有明显锯齿。 |
| SWS_AREA | 114 | 边缘有不明显锯齿。 |
| SWS_BICUBLIN | 95 | 与上上上图几乎无区别。 |
| SWS_GAUSS | 86 | 比上图边缘略微清楚一点。 |

| | | |
|---|---|---|
| SWS_SINC | 20 | 与上上图无区别。 |
| SWS_LANCZOS | 64 | 与上图无区别。 |
| SWS_SPLINE | 40 | 与上图无区别。 |

总评，Point算法有明显锯齿，Area算法锯齿要不明显一点，其余各种算法，肉眼看来无明显差异。此外，使用CImage进行渲染时缩放，帧率可达105，效果与Point相似。

个人建议，如果对图像的缩放，要追求高效，比如说是视频图像的处理，在不明确是放大还是缩小时，直接使用SWS_FAST_BILINEAR算法即可。如果明确是要缩小并显示，建议使用Point算法，如果是明确要放大并显示，其实使用CImage的Strech更高效。

当然，如果不计速度追求画面质量。在上面的算法中，选择帧率最低的那个即可，画面效果一般是最好的。

不过总的来说，ffmpeg的scale算法，速度还是非常快的，毕竟我选择的素材可是高清的图片。

(本想顺便上传一下图片，但各组图片差异其实非常小，恐怕上传的时候格式转换所造成的图像细节丢失，已经超过了各图片本身的细节差异，因此此处不上传图片了。)

注：试验了一下OpenCV的Resize效率，和上面相同的情况下，OpenCV在上面的放大试验中，每秒可以进行52次，缩小试验中，每秒可以进行458次。

原文地址： http://www.cnblogs.com/acloud/archive/2011/10/29/sws_scale.html

更新（2014.8.5）=================

FFmpeg使用不同sws_scale()缩放算法的命令示例（bilinear，bicubic，neighbor）：

```plain
1. ffmpeg -s 480x272 -pix_fmt yuv420p -i src01_480x272.yuv -s 1280x720 -sws_flags bilinear -
   pix_fmt yuv420p src01_bilinear_1280x720.yuv
2. ffmpeg -s 480x272 -pix_fmt yuv420p -i src01_480x272.yuv -s 1280x720 -sws_flags bicubic -pix_fmt yuv420p src01_bicubic_1280x720.yuv
3. ffmpeg -s 480x272 -pix_fmt yuv420p -i src01_480x272.yuv -s 1280x720 -sws_flags neighbor -
   pix_fmt yuv420p src01_neighbor_1280x720.yuv
```

文章标签： ffmpeg sws_scale 性能

个人分类： FFMPEG

所属专栏： FFmpeg