基于 API 的图像处理算法的快速实现①

刘胜达 [1,2] 单晓光 [1] (1.哈尔滨德强商务学院 黑龙江 哈尔滨 150025;
2.哈尔滨理工大学 计算机科学与技术学院 黑龙江 哈尔滨 150080)

摘 要: 介绍了如何利用 API 快速实现图像处理算法。首先通过对几种常见图像处理算法的实现方法进行分析,发现运用 API 函数可对图像处理算法进行快速实现,再提出几种更快的利用 API 处理图像的方法。结果表明利用 API 函数能快速有效的处理图像,且具有很好的应用价值。

关键词: 图像处理; 算法; 分析; 优化; 快速实现

Realization and Optimization of Image Processing Algorithm Based on API

LIU Sheng-Da[1,2], SHAN Xiao-Guang[1]
(1. Harbin Deqiang College Of Commerce, Harbin 150025, China; 2. College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China)

Abstract: The fast realization of image processing based on API is discussed in this paper. Firstly a few familiar methods of image processing are analyzed. It is discovered that the image processing could be carried out rapidly by using API function, and a few rapid image processing methods are put forward. The result shows that the image processing can be achieved rapidly and effectively through the API function. And it has better practical value.

Keywords: image processing; algorithm; analyzed; optimization; fast realization

1 引言

图像处理方面的软件和算法很多,但有时为了方便快速的在自己的程序中直接处理图像,就需要自己编写图像处理程序[1]。下面就针对几类常见的处理图像方法进行实现和优化,同时也希望有兴趣的读者可以和本文作者共同探讨。

2 程序执行时间检测算法

算法效率的度量通常有两种方法,分别是事前分析估算法和事后统计法[2]。事前分析估算法较事后统计法误差大,因此本文采用事后统计法。

可在按钮事件开始时调用 API 函数 GetTickCount()获得从操作系统启动到现在所经过的毫秒数[3],在按钮事件结束时再调用一次 GetTickCount(),然后做差。实现方法如下:

```
Private Sub Command1_Click()
t1 = GetTickCount()
……
t2 = GetTickCount()
Print t2 - t1
End Sub
```

运行程序十次,然后取其平均运行时间。

3 图像处理程序算法的实现及优化

算法就是一段程序,该程序段对给定的输入可在有限的时间内产生出确定的输出结果,算法可采用多种语言来描述[4]。API 结合 VB 优越的图形界面[5]能有效的实现图像的快速处理,因此本文利用 VB 来描述

① 基金项目:哈尔滨德强商务学院基金
收稿时间:2009-09-05

图像处理算法，并探讨如何利用 Windows API 来进行图像处理算法的实现及优化。

3.1 图像旋转和反转的实现及优化

3.1.1 算法的普通实现

双重循环遍历每个像素点。循环体为：

```
Picture2.PSet (j, Picture1.ScaleWidth - i - 1), Picture1.Point(i, j)
```

该方法可实现图像的左旋 90 度。

3.1.2 优化

对于 9 万像素的图像算法的普通实现方法平均运行时间为 1200 毫秒，这显然有些长了。优化方法如下：

```
SetPixelV Picture2.hdc, j, Picture1. ScaleWidth-i-1, Picture1.Point(i, j)
```

该方法调用 API 函数 SetPixelV 来实现，其平均运行时间为 867 毫秒，但还是较慢。可再对其进行优化，方法如下：

```
SetPixelV Picture2.hdc, j, Picture1.ScaleWidth-i-1, GetPixel(Picture1.hdc, i, j)
```

该方法调用 API 函数 SetPixelV 和 GetPixel 来实现，其平均运行时间为 766 毫秒，还是不够快。更快的实现方法如下：

```
pt3(0).x = 0
pt3(0).y = Picture2.Height - 4
pt3(1).x = 0
pt3(1).y = 0
pt3(2).x = Picture2.Width - 4
pt3(2).y = Picture2.Height - 4
PlgBlt Picture2.hdc, pt3(0), Picture1.hdc, 0, 0, Picture1.Width - 4, Picture1.Height - 4, 0, 0, 0
```

Picture2.Refresh’刷新 Pictrue2，每次修正优化后都有此句，本文以下部分将其省略。

该方法调用 API 函数 PlgBlt 来实现，其平均运行时间为 46 毫秒，较算法的普通实现方法提高速度近 30 倍。

同理，用此方法可实现图像的右旋 90、水平反转、垂直反转，较算法的普通实现方法提高速度近 30 倍。

特别的，用此方法实现图像的旋转 180 度仅需 9 毫秒，较算法的普通实现方法提高速度近 150 倍。

3.2 图像去色的实现及优化

3.2.1 算法的普通实现

双重循环遍历每个像素点。循环体为：

```
col = Picture1.Point(i, j)
……
Picture2.PSet (i, j), RGB(pj, pj, pj)’pj 为原 RGB 平均值
```

3.2.2 优化

对于 9 万像素的图像算法的普通实现方法平均运行时间为 1134 毫秒，速度显然不够快。优化方法如下：

```
col = GetPixel(Picture1.hdc, i, j) ‘替换 3.2.1 中第一句
SetPixelV Picture2.hdc, i, j, RGB(pj, pj, pj) ‘替换第 3.2.1 节中最后一句
```

该方法调用 API 函数 GetPixel 和 SetPixelV 来实现，其平均运行时间为 707 毫秒，虽然速度有所提高，但还是不够快。更快的实现方法如下：

```
GetDIBits Picture1.hdc, Picture1. Picture. Handle, 0, bi32bitinfo.bmiheader.biheight, bby(1), bi32bitinfo, 0
For i = 1 To bi32bitinfo.bmiheader.biwidth * bi32bitinfo.bmiheader.biheight Step 1
r = bby(i).rgbred
g = bby(i).rgbgreen
b = bby(i).rgbblue
y = (r + g + b) / 3
bby(i).rgbred = y
bby(i).rgbgreen = y
bby(i).rgbblue = y
Next i
SetDIBitsToDevice Picture2.hdc, 0, 0,
bi32bitinfo.bmiheader.biwidth,
bi32bitinfo.bmiheader.biheight, 0, 0, 0,
bi32bitinfo.bmiheader.biheight, bby(1),
bi32bitinfo, 0
```

该方法调用 API 函数 GetDIBits 和 SetDIBits-ToDevice 来实现，其平均运行时间为 128 毫秒，较算法的普通实现方法速度提高近 9 倍。

该方法稍加修改既可实现图像的锐化、柔化、钝化和浮雕等，较算法的普通实现方法均提高几倍的速度。

3.3 图像反色的实现及优化

3.3.1 算法的普通实现

双重循环遍历每个像素点。循环体为：

```
col = Picture1.Point(i, j)
……
Picture2.PSet (i, j), RGB(255 -r, 255 - g, 255 - b)
```

3.3.2 优化

对于 9 万像素的图像算法的普通实现方法平均运行时间为 1129 毫秒，速度显然不快。优化方法如下：

```
col = GetPixel(Picture1.hdc, i, j) '替换 3.3.1 中第一句
SetPixelV Picture2.hdc, i, j, RGB(255 - r, 255 - g, 255 - b) '替换 3.3.1 中最后一句
```

该方法调用 API 函数 GetPixel 和 SetPixelV 来实现，其平均运行时间为 659 毫秒，虽然速度提高近一倍，但还是不够快。更快的实现方法如下：

```
BitBlt Picture2.hdc, 0, 0, Picture2.Width,
Picture2.Height, Picture1.hdc, 0, 0,
NOTSRCCOPY
```

该方法调用 API 函数 BitBlt 来实现，其平均运行时间不到 1 毫秒，瞬间实现了图像的反色处理。也可用如下方法实现。

```
rect1.Bottom = Picture2.Height
rect1.Left = 0
rect1.Top = 0
rect1.Right = Picture2.Width
Picture2.Picture = Picture1.Picture
InvertRect Picture2.hdc, rect1
```

该方法调用 API 函数 InvertRect 来实现，其平均运行时间不到 1 毫秒，同样瞬间实现了图像的反色处理。

3.4 图像半透明的实现及优化

3.4.1 算法的普通实现

双重循环遍历每个像素点。循环体为：

```
col1 = Picture1.Point(i, j)
……
col2 = Picture2.Point(i, j)
……
Picture2.PSet (i, j), RGB(r, g, b)
```

3.4.2 优化

对于 9 万像素的图像算法的普通实现方法平均运行时间为 1572 毫秒，速度显然很慢。优化方法如下：

```
col1 = GetPixel(Picture1.hdc, i, j) '替换 3.4.1 的第一句
col2 = GetPixel(Picture2.hdc, i, j) '替换 3.4.1 的中间一句
SetPixelV Picture2.hdc, i, j, RGB(r, g, b) '替换 3.4.1 的最后一句
```

该方法调用 API 函数 GetPixel 和 SetPixelV 来实现，其平均运行时间为 1031 毫秒，速度提高不多，依然很慢。更快的实现方法如下：

```
AlphaBlend Picture2.hdc, 0, 0,
Picture1.ScaleWidth, Picture1.ScaleHeight,
Picture1.hdc, 0, 0, Picture1.ScaleWidth,
Picture1.ScaleHeight, &H7F0000
```

该方法调用 API 函数 AlphaBlend 来实现，其平均运行时间不到 1 毫秒，瞬间实现了图像的半透明处理。

3.5 马赛克的实现及优化

3.5.1 算法的普通实现

```
col = Picture1.Point(k, l)
……
Picture2.PSet (k, l), RGB(r, g, b)
```

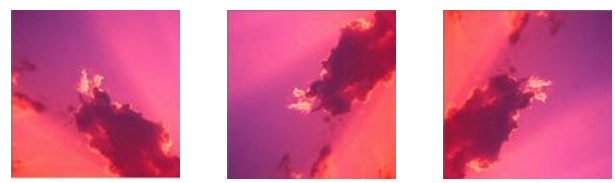
3.5.2 优化

对于 9 万像素的图像算法的普通实现方法平均运行时间为 1033 毫秒，速度显然不够快。可采用 API 函数 GetPixel 和 SetPixelV 来优化，其平均运行时间为 682 毫秒，速度还是不理想。更快的实现方法如下：

```
For i = 0 To Picture1.ScaleWidth Step 5
  For j = 0 To Picture1.ScaleHeight Step 5
  For m = 0 To 3
  BitBlt Picture2.hdc, i + m, j, 5, 5,
  Picture1.hdc, i + 2, j + 2, SRCCOPY
  Next m
  Next j
Next i
```

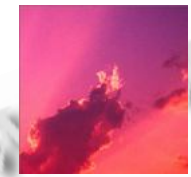
该方法调用 API 函数 BitBlt 来实现，其平均运行时间为 89 毫秒，较算法的普通实现方法提高速度 10 多倍。另外，还可以通过改变循环体步长来优化，笔者曾优化到平均运行时间为 15 毫秒。

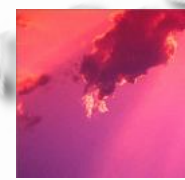
3.6 效果图

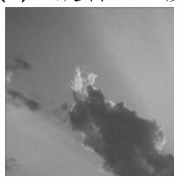


(a) 原始图像 (b) 左旋 90 度 (c) 右旋 90 度


(d) 旋转180度


(e) 水平反转


(f) 垂直反转


(g) 去色


(h) 反色


(i)半透明


(j) 马赛克

图 1 效果图

4 结语

本文介绍了基于 API 利用 VB 进行图像处理的几类常见算法的快速实现，结果表明，巧妙运用 API 函数，能大大提高处理图像的速度。其实，利用 API 同样可以大大加快正交变换、图像增强、图像复原和图像分割等处理速度。

参考文献

1 李俊荣,王振明.用VB编写图像处理程序算法的实现.计算机应用与软件, 2007,(9):215－217.
2 严蔚敏.吴伟民,数据结构.北京:清华大学出版社,1997.
3 范文庆,周彬彬,安靖.精通 Windows API:函数、接口、编程实例.北京:人民邮电出版社, 2009.
4 胡学钢.数据结构,北京:高等教育出版社, 2008.
5 高西宽,刘泊,马熙源.基于 VB 与 MATLAB 混合编程的数字水印软件设计.哈尔滨理工大学学报, 2008,(8):17－20.