Bresenham 画圆法提取图像信息①

卫洪春

(四川文理学院 计算机科学系, 达州 635000)

摘 要: 在图像处理中, 若操作图像某区域中的像素, 必须先提取这些待处理的像素. 待处理的像素区域可以是规则的, 也可能是不规则的. 对于规则区域, 可采用经典的图形学算法完成选取; 对于不规则区域, 可通过适当方法将其转换为规则区域来完成选取. 讨论了 24 位真彩色图像文件的结构、位图数据阵列中的像素与其在屏幕上显示的像素间的关系; 采用 Bresenham 画圆法提取图像圆形区域中的像素并将其可视化等问题.

关键词: 位图; 结构; Bresenham 画圆; 矩阵; 提取

Extracting Image Information by Bresenham Drawcircle

WEI Hong-Chun

(Department of Computer Science, Sichuan University of Arts and Science, Dazhou 635000, China)

Abstract: During image processing, if the pixels in some region of the image are operated, those pixels to be processed must be extracted. The area including pixels to be processed may be regular, and may also be irregular. As for regular area(for example, polygon, rectangle, circle and ellipse), it can be picked up by classical graphics algorithm; if the area is irregular, it can be converted into regular area by appropriate method. This paper discusses the following problems: the structure of 24-bit true color BMP file; the relationship between pixels in bmp data array and pixels to be showing on screen; extracting pixels, which belong to circle area of image, by Bresenham drawcircle; the visibility of those pixels be extracted.

Key words: BMP; structure; Bresenham drawcircle; arry; extracting

在图像处理过程中, 若需处理一幅 24 位真彩色图像中的一些像素, 首先需将该图像读入计算机, 然后用某种方法选取该图像中待处理像素, 最后完成处理. 图像像素区域的获取对于像素的后续处理至关重要, 若不能得到待处理的像素, 便无法完成图像的处理. 因此, 必须从读入的图像中获取待处理的各个像素在图像中的位置及其对应的颜色值. 本文探讨了基于 Bresenham 画圆法的图像信息提取技术.

1 位图文件的结构

BMP 图像文件格式是 Windows 环境中交换与图像有关的数据的一种标准. 24 位真彩色 BMP 结构包括 3 部分: 位图文件头 BITMAPFILEHEADER、位图信息头 BITMAPINFOHEADER、位图数据阵列[1].

1.1 文件头 BITMAPFILEHEADER 的结构

文件头 BITMAPFILEHEADER 的结构定义及含义如下:

```
typedef struct tagBITMAPFILEHEADER{
WORD bfType; // 位图文件类型是”BM”(1-2 字节)
DWORD bfSize; // 位图文件的大小 (3-6 字节)
WORD bfReserved1; //保留字, 必须为 0(7-8 字节)
WORD bfReserved2; //保留字, 必须为 0(9-10 字节)
DWORD bfOffBits;  //位图数据的起始位置相对于
                  //图文件头的偏移量 (11-14 字节)
} BITMAPFILEHEADER;
```

1.2 位图信息头 BITMAPINFOHEADER 结构

位图信息头 BITMAPINFOHEADER 结构用于说明位图的尺寸等信息, 其定义如下:

① 基金项目:国家自然科学基金(61152003)
收稿时间:2013-05-31;收到修改稿时间:2013-07-24

```
typedef struct tagBITMAPINFOHEADER{
DWORD biSize; // 本结构占用字节数(15-18 字节)
LONG biWidth; //以像素表示的位图宽度(19-22 字节)
LONG biHeight;//以像素表示的位图高度(23-26 字节)
WORD biPlanes; //目标设备级别必须为 1(27-28 字节)
WORD biBitCount;//各像素所需的位数: 1(双色),
    //4(16 色), 8(256 色)或 24(真彩色) (29-30 字节)
    DWORD biCompression; //压缩类型: 0(不压缩),
    //1(BI_RLE8 压缩),2(BI_RLE4 压缩) (31-34 字节)
    DWORD  biSizeImage;//以字节表示的位图大小
(35-38 字节)
    LONG biXPelsPerMeter; //位图水平分辨率,
                          //每米像素数(39-42 字节)
    LONG biYPelsPerMeter; //位图垂直分辨率,
                          //每米像素数(43-46 字节)
    DWORD biClrUsed;
    //位图实际使用的颜色表中的/颜色数(47-50 字节)
    DWORD biClrImportant;
        // 位图显示中重要的颜色数(51-54 字节)
    } BITMAPINFOHEADER[2,3];
```

1.3 位图数据阵列

位图文件中的位图数据记录了位图中的每个像素值. 位图像素数据从文件第 55 字节开始, 每个像素占 3 个字节, 这 3 个字节从低到高依次表示该像素的蓝色 B、绿色 G、红色 R 分量的亮度值. 文件中记录的像素顺序是: 在扫描行内按从左到右的像素顺序记录,扫描行之间按照从下到上的顺序记录, 图像数据阵列中的第一个字节表示位图左下角的象素, 而最后一个字节表示位图右上角的象素.

位图数据阵列中每行记录的像素值与其显示在屏幕上的像素的对应关系如图 1 所示.

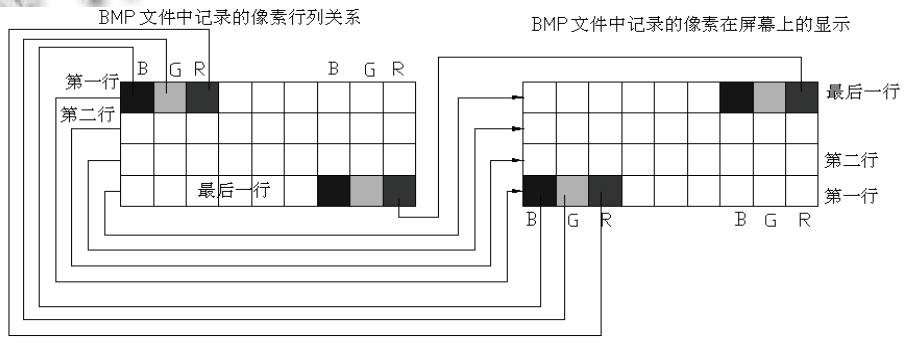

图 1 位图数据阵列中的各像素与其显示在屏幕上的像素的对应关系

2 Bresenham画圆算法

首先考虑圆心位于点 O(0, 0)、半径为 r 的圆从 x=0、y=r 为起始点, 沿顺时针方向生成 1/8 圆弧的过程. 设(x, y)是圆 $x^2+y^2=r^2$ 上的点, (x_i, y_i)是生成圆上的点. x_i 的取值始于 $x_i=x=0$, 以步长为 1 增至点 x=y 所对应的 x 坐标, 且为整数. 当 $x_{i+1} = x_i + 1$ 时, 计算其生成圆上对应点 (x_{i+1}, y_{i+1})的坐标分量 y_{i+1}. 若函数值 y 相对于 $y_i - 1$ 更靠近 y_i, 则 $y_i+1=y_i$; 否则 $y_{i+1} = y_i - 1$, 如图 2 所示. 计算过程如下:

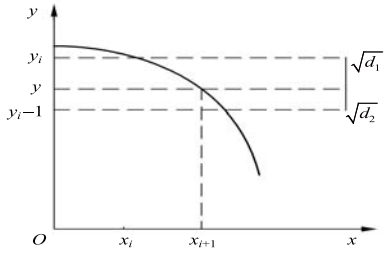

图 2 Bresenham 画圆算法

$$y^2 = r^2 - (x_i + 1)^2$$

$$d_1 = y_i^2 - y^2 = y_i^2 - r^2 + (x_i + 1)^2$$

$$d_2 = y^2 - (y_i - 1)^2 = r^2 - (x_i + 1)^2 - (y_i - 1)^2$$

令误差项 $p_i = d_1 - d_2$, 则:

$$p_i = d_1 - d_2$$

$$= 2(x_i + 1)^2 + y_i^2 + (y_i - 1)^2 - 2r^2 \quad (a)$$

若 $p_i<0$, 则 $y_{i+1} = y_i$; 否 则 $y_{i+1} = y_i - 1$

$$p_{i+1} = 2(x_{i+1} + 1)^2 + y_{i+1}^2 + (y_{i+1} - 1)^2 - 2r^2 \quad (b)$$

由(b) - (a)且 $x_{i+1} = x_i + 1$得:

$$p_{i+1} - p_i = 4x_i + 6 + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i) \quad (c)$$

故 p_i的递归式为:

$$p_{i+1} = p_i + 4x_i + 6 + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i) \quad (d)$$

确定 p_i的初值: 将 $x_1 = 0$, $y_1 = r$代入(a)式, 得

$$p_1 = 3 - 2r \quad (e)$$

根据 Bresenham 画圆算法, 利用圆的对称性可完成全圆的绘制, 如图 3 所示. 在图 3 中, XOY 是直角坐标系, OXs-Os-OYs 是屏幕坐标系. 在直角坐标系 XOY 中, 采用 Bresenham 画圆法, 当计算出图 3 中的点 A1 的坐标为(x, y)后, 根据对称性可依次计算出其余各点的坐标: A2(y, x)、A3(y, -x)、A4(x, -y)、A5(-x, -y)、A6(-y, -x)、A7(-y, x)、A8(-x, y)[4,5].

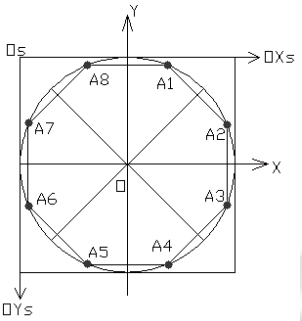

图 3　对称点画圆算法

若要获取图像圆形区域中的各个像素, 需将待绘制的圆所包含的区域与待选取的圆形图像区域形成一一映射关系. 映射原理如下:

设待绘制圆的圆心 O 点的屏幕坐标是(x, y), 半径为 r, 由此可得到该圆的外接正方形(正方形的边与坐标轴平行). 该正方形中各个像素点的坐标可存储在一个二维矩阵 p 中. 二维矩阵 p 各像素的布局与屏幕坐标系一致. 先将该矩阵中的各个元素赋初值-1, 矩阵的行列号表示该像素的直角坐标分量. 当移动鼠标时, 可根据不断变化的圆的半径 r 动态生成正方形所对应的二维矩阵 p. 当释放鼠标左键后, 即可确定该圆的半径, 此时, 正方形所对应的二维矩阵 p 也被确定. 然后采用 Bresenham 画圆算法, 设置圆边界点所对应的二维矩阵中的相应元素的值为 1, 而二维矩阵 p 中圆边界没有经过的点所对应的矩阵的元素的值不变, 仍为-1. 绘制圆时, 若某元素的值为 1, 则在屏幕上该元素行列号所对应的像素点处绘制一个像素; 若某元素的值为-1, 则在屏幕上该元素行列号所对应的位置处不绘制该像素.

由于圆的半径是 r, 故二维矩阵 p 的大小是(2*r+1)*(2*r+1). p[0][0]元素对应正方形的左上角点. 直角坐标系中 A1(x, y)点所对应的像素在二维矩阵中的元素是(r-y, r+x), 每行 2r+1 个点, 直角坐标系中的第 y 行在屏幕坐标系中是第 r-y 行.

3　算法设计与实现

在 VC6.0 环境下, 新建一基于 MFC 的单文档工程[6].

3.1　设计基于 Bresenham 画圆法的圆类

在该项目中添加 PickCircle 类[7]. 该类实现对图像圆形区域的选取, 类的成员如表 1 所示.

表 1　PickCircle 类

PickCircle
int xc, yc;　//圆心坐标 int r;　//圆的半径
void BresenhamDraw(CDC *pDC,int* pCircle); // 选取圆形区域，置圆的边界的标识为 1

选取圆形区域的 Bresenham 画圆算法[8-11]如下:

(1) 置当前像素为(0, r), 置误差初值 p_1=3-2r, i=1

(2) 置当前像素及其七个对称点标识为 1(见图 3)

(3) 计算下一个当前点(x_{i+1}, y_{i+1}). 其中 $x_{i+1}=x_i+1$. 若 $p_i<0$ 则 $y_{i+1}=y_i$, 且下一个误差 $p_{i+1}=p_i+4*x_i+6$; 否则 $y_{i+1}=y_i-1$, 且下一个误差为 $p_{i+1}=p_i+4*(x_i-y_i)+10$.

(4) i=i+1, 若 $x_i=y$ 则结束, 否则返回(2).

3.2　视图类 View 的设计与实现

完成拾取图像圆形区域的圆类 PickCircle 的设计后, 主要任务是处理视图类 View. 详细的处理过程如下.

(1) 在视图类 View 中添加表 2 中的成员变量.

表 2　视图类 View 中添加的数据成员

数据类型	变量名称	说明
long	lineTotal	每行像素数, 对齐方式是 DWORD 对齐
char	r,g,b	像素的 RGB 分量
int	MapWidth, MapHeight	位图宽、高
char*	pixelRGB	指向位图数据阵列的指针
char*	pixelRGBTrans	指向转换后的位图数据的指针, 即将位图的第 i 行与第 MapHeight-i 行对换后的位图数据.
int*	pCircle	指向包围圆形的最小正方形区域的指针
int	currentPx, currentPy	鼠标当前点的 x、y 坐标
bool	flagDraw	绘图标识
PickCircle	pc	圆类对象,获取圆的边界点

(2) 在视图类 View 的构造函数中加载位图文件, 并将位图文件中的位图数据阵列存放在指针变量 pixelRGB 所指向的空间, 算法[12]如下:

初始化绘图标识
初始化指向包含圆形的正方形区域指针为 NULL
打开位图文件
填充位图文件头结构, 填充位图信息头结构
获取位图数据阵列的高度
获取位图数据阵列的宽度(每行的像素数)
获取位图数据阵列总的字节数
获取位图数据阵列每行的字节数
动态生成存储位图数据阵列的空间
获取位图数据并关闭文件
动态正向放置位图数据的空间
将位图数据正向放置 //以与屏幕坐标系一致.

(3) 在 View 类的析构函数中释放动态生成的内存空间 pixelRGB 及 pixelRGBTrans.

(4) 在 View::OnDraw(CDC* pDC)函数中动态显示待绘制的圆形区域的半径, 用以动态显示随鼠标移动时, 即时将生成的圆显示在屏幕上.

(5) 添加单击鼠标左键的响应函数 View::OnLButtonDown(). 当按下鼠标左键时, 记录圆心坐标, 重置绘图标识 flagDraw 为真, 开始绘制圆, 该响应函数的设计如下:

初始化圆 pc 的圆心坐标分量 xc 及 yc
flagDraw = true; //重置绘图标识

(6) 添加响应鼠标移动的函数, 在鼠标移动过程中动态获取圆的半径, 同时动态生成指向包含该圆的正方形所对应的二维矩阵. 为方便处理, 将二维矩阵以与其等价的一维矩阵来表示. 函数 View::OnMouseMove()的伪代码描述如下:

用当前鼠标点的屏幕坐标 currentPx、currentPy.
计算当前鼠标点与包围圆心(xc,yc)之差 dx,dy.
计算包围圆 pc 的半径 $r=\sqrt{dx^2+dy^2}$
若当前处于绘图状态{
计算包围圆 pc 的最小正方形所需的像素数
若指向正方形区域的指针非空, 则删除
重新生成正方形区域
重置正方形区域中的每个像素为-1 //-1 为不绘制
刷新 }

(7) 添加鼠标左键释放函数 View::OnLButtonUp(), 释放鼠标左键后在屏幕上对应位置绘制选定的图像圆形区域的像素, 该算法的设计如下:

设置绘制圆的标识为不绘制:flagDraw = false
重绘窗口, 获取当前设备上下文
若指向正方形区域的指针非空且不处于绘图状态{
调用 Bresenham 画圆法, 置圆的边界像素点为 1

```
for(int i = 0; i < 2*pc.r+1; i++)     //控制行
for(int j = 0; j < pc.r+1; j++){//控制列到圆的正中
if(pCircle[i*(2*pc.r+1)+j] == 1){
for(int k=j;k< 2*pc.r+1-j;k++){
 if(i <=MapHeight ){
pDC->SetPixel(//绘制指定像素
k+pc.xc-pc.r,   i+pc.yc-pc.r, RGB(
   pixelRGBTrans[(i+pc.yc-pc.r)*lineTotal
                 +3*(k+pc.xc-pc.r)+2], //红色 R
   pixelRGBTrans[(i+pc.yc-pc.r)*lineTotal
                 +3*(k+pc.xc-pc.r)+1], //绿色 G
   pixelRGBTrans[(i+pc.yc-pc.r)*lineTotal
                 +3*(k+pc.xc-pc.r)] )   //蓝色 B
);} } } } }
```

4 结论

基于 Bresenham 画圆法的图像信息提取的测试结果如图 4 所示. 根据实际情况, 待选取的图像像素区域除了圆形外, 还可能是其它形状的平面图形(如矩形、不规则图形等). 尽管实际情况比圆形区域复杂, 但可以借助本问题的思路, 完成对给定图像区域像素的选取, 为后续的图像处理获取待处理的数据.

(a) 原始图片

(b) 提取圆形区域像素

图 4 测试结果

参考文献

1 赵君,王乘.图像格式分析与图像显示实现.计算机与数字工

(下转第 27 页)

态, 可将 HTTP 请求转交给相应的 JSP 页面, 并发送响应到浏览器端, 以显示用户行为的采集结果、用户行为数据的预处理结果、用户行为序列模式的挖掘结果和网站管理决策的描述.

6 结语

在本文所提出的一种基于 Struts2 技术的社交网络服务平台设计方案中, 笔者已对社交网络服务平台的用例模型、架构、设计模式、应用技术以及用例实现过程进行了阐述和分析. 该设计方案的提出进一步体现了面向对象程序设计的灵活性和可重用性, 已将具有模型层、视图层和控制层的平台结构进行了清晰的逻辑划分, 已将业务逻辑、显示逻辑和控制逻辑进行了有效的分离, 在满足企业级应用复杂性和安全性需要的同时, 也充分体现了基于 Struts2 技术的网络应用服务平台设计思想已成为一种主流的发展态势.

参考文献

1 Rémy MW, Michiko Y, Tomoaki W. Social network productivity in the use of SNS. Journal of Knowledge Management, 2010, 14(6): 910–912.

2 Babushkina YV. Using web 2.0 for the information support of employees. Scientific and Technical Information Processing, 2011, 38(1): 38–40.

3 李美子,张波.社交网络中的用户信任链形式化模型.计算机工程,2012,38(23):60.

4 黄建远.SNS 社交网站电子商务经营研究.现代传播,2012,(3):161.

5 蔡骥然,曹海传.B/S 架构下基于 OPC 与 Comet 技术的实时监控系统.计算机应用,2012,32(S2):215.

6 Hao LC. Application of MVC platform in bank E-CRM. International Journal of Service, Science and Technology, 2013, 6(2): 34–36.

7 Li YF, Chen ZG. Design and implement of news publishing system based on MVC design pattern. Advances in Intelligent Systems and Computing, 2013, 181: 756–757.

8 Chen LY, Chen J, Xu B. The implement of AJAX in Struts2 framework. International Review on Computers and Software, 2012, 7(6): 3206–3207.

9 Tian J, Xu B, Chen L. Design and implementation of BLOB data processing mechanism based on struts2 and hibernate. International Journal of Advancements in Computing Technology, 2012, 4(14): 77–78.

10 李红,吕本富,申爱华.SNS 网站竞争生存及商业模式创新的关键因素实证研究.管理评论,2012,24(8):80.

(上接第 131 页)

程,2004,32(5):1–4.

2 王海欣,邓中亮.位图图象的读取及基本操作.计算机应用研究,2001,18(2):17–18.

3 耿卫东,陈为.计算机游戏程序设计.第 2 版.北京:电子工业出版社,2009.

4 孙家广,许隆文.计算机图形学.北京:清华大学出版社,1998.

5 杜晓增,丁宇辰.计算机图形学基础.北京:机械工业出版,2004.

6 沈荣,廖婷.图形对象拾取技术在开发 CAD 系统中的应用.四川文理学院学报,2012,22(5):72–76.

7 郑莉,董渊,何江舟.C++语言程序设计.第 4 版.北京:清华大学出版社,2010.

8 丁宇辰.圆弧的生成算法研究.南京工程学院学报(自然科学版),2010,8(2):59–62.

9 孙崇璇.圆弧生成方法探究与实现.云南师范大学学报(自然科学版),2011,31(4):64–68.

10 生鸿飞,庞爱民.圆弧生成的 C 程序算法研究.武汉科技学院学报,2007,20(3):1–3.

11 王志喜,王润云.Bresenham 画圆算法的改进.计算机工程,2004,30(12):178–180.

12 胡明星.在 VC 下进行图象处理的方法.计算机工程,2000,16(10):91–95.