二值图像噪声控制的算法实现
Algorithms of Image Noise Control in Bitonal Image

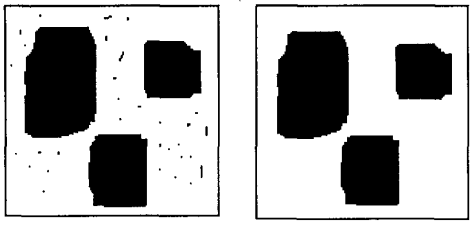
卫 锋
Wei Feng
(广西计算中心,广西南宁 530022)
(Guangxi Computing Centre,Nanning,Guangxi,530022,China)

摘要:介绍二值图像噪声控制的实现思路,给出二值图像噪声控制实现的2个方法:种子填充算法和扫描线算法,分析比较2个算法实现的效率。结果表明,扫描线算法的效率比种子填充算法高。
关键词:二值图像噪声 种子填充算法 扫描线算法
中图法分类号:TP391
Abstract: The approaches of image noise control are introduced. Two algorithms of the seed fill and the scan-line are discussed in control of image noise. The scan-line is better than the seed fill in efficiency.
Key words: bitonal image noise, the seed fill algorithm, the scan-line algorithm

近年来图像扫描的需求急剧增加。二值图像因存储体积小被广泛应用于各类图像扫描。二值图像文件只保留黑白两色,在将纸质资料扫描成为二值图像后会在图像上产生许多杂点。为提高图像的质量,需要将这些杂点去除,即图像去噪。图像不大时这种去噪不会有太多问题,而对大图像时必须考虑其实现的效率问题。图像去噪时,真彩图像、灰度图像和二值图像的方法不尽相同,二值图像的噪声去除多采用删除噪声点的方法。本文给出目前二值图像处理算法中流行的种子填充算法和扫描线算法的实现方法,比较分析这2个算法实现的效率。

1 实现思路

首先将图像中面积不超过一个常数的区域填充为指定的颜色,处理前后的图像噪声控制效果如图1所示。



(a) (b)

图1 图像噪声控制效果对比
(a)处理前;(b)处理后

本文先作一些定义:采用一个二维数组来定义一个图像:char[i][j],二值图像中使用0代表黑色、1代表白色。图像的规则采用四连通区域规则,即一个点的上下左右是相互连通的,而像素点的对角连线则不视为连通[1]。将具有相同颜色并且连通的点集称为一个连通区域,区域中像素点的个数为该区域的面积。要求将一个图像中面积不超过C的黑色区域填充为白色。

本文实现的思路为:扫描整个数组,对每个点进行判断,如果遇到黑色的点则对其连通的点进行判断,如果连通的点的数量小于C,则对其值进行重新设置。

2 两种算法实现

目前图像区域填充算法使用较多的是种子填充算法和扫描线填充算法,这两种算法各有利弊[1],而实现的方法也有很多种,现在就这两种算法二值图像噪声控制的实现进行描述。

2.1 种子填充算法的实现

该算法首先遍历图片数组,得到一个黑色的点,然后开始递归遍历相邻的像素点,并且设置一个变量计算连通区域的面积,然后根据计算出来的面积判断是否要填充,如果需要填充则立即填充,不需要填充的区域最后统一填充为黑色。

以下是使用C语言表示的部分代码:

```
Image[M][N]; //图片数组
Int area;//记录面积变量
for(i=0;i<M;i++)
```

2004-09-18 收稿。

```
{
  for(j=0;j<N;j++)  //遍历数组
  {
    area = 0;  //记录面积变量置 0
    scanImage(i,j)  //计算面积
    if (area<C)  //如果面积小于 C 则进行填充
    {
        fillImage (i,j)  //填充图片
    }
  }
}
  //再将不符合要求的区域填充成黑色
for( i=0; i<M; i++ )
{
  for(j=0;j<N;j++)
  {
    if(Image[i][j]==-1)
      Image[i][j]=0;
  }
}
scanImage(i,j)//计算面积函数
{
  if (Image[i][j] ! =-1)  //表示改像素点没有被扫描过
  {
    Image[i][j]=-1;//标识该像素点并且面积增加
    area++;
    //递归搜索区域
    scanImage(i,j+1);
    scanImage(i,j-1);
    scanImage(i+1,j);
    scanImage(i-1,j);
  }
}
```

该算法搜索到一个未搜索的黑色点时，将其数组设置为－1，以标识已经搜索过，避免重复搜索。而搜索的面积小于 C 时则立即填充该区域，而后将所有的值为－1 的像素点回填为黑色，进一步提高效率。由于采用了递归算法，当遇到像素点很多的大图像时递归的效率将严重的下降，而且该算法采用了两次填充。

2.2 扫描线算法的实现

扫描线算法的实现方法即是采用一行一行的扫描，一行中连续的黑色区域定义为一个 piece，而一个连续的 piece 定义为一个 area。当 area 不再延续并且面积大于 C 时即清除这个 area，而当 area 不再延续但是面积小于 C 时填充所有的 piece 并且清除 area。

具体的算法实现流程采用伪语言方式描述如下：

```
先按照像素进行处理
扫描一行中的所有像素
if 像素为黑色
  if 前一像素为白色 then  //搜索到一个新的 piece
    记录新的 piece 的起始位置
  end if
  if 当前的 piece 不属于任何 area then
    if 上一列像素对应的 area then
      设置上一个 area 延续
      则设置当前 piece 中所有像素对应的 area
    end if
  end if
else 像素为白色
  如果前一像素为黑色则设置 piece 的结束位置
end if
if 最后一个像素为黑色 then
  记录最后一个 piece 结束的位置
end if
然后处理一行中所有的 piece
if piece 属于一个 area
  区域的面积增加
  如果加入后区域面积<C
  将 piece 加入 area
else piece 不属于一个 area
  创建一个新的 area 并且将其加入链表中
end if
下面检查所有的 area
if area 延续到正在处理的行
  设置连续标志
else area 不延续
  if area 的面积<C then
    填充 area 中所有的 piece
```

end if

清除该 area 中所有的 piece 并且清除该 area

end if

扫描线算法用来记录一个区域的轮廓，进而来判断一个区域的面积。通过记录每一行黑色区域的起点和终点来存储黑色区域的范围和计算黑色区域的面积，而黑色块面积采用计算连通的黑色区域的面积得到。每行黑色区域和总黑色块面积采用结构体来存储。如果面积小于C则立即清除和删除相关资源。该算法通过一行一行的检测避免了重复检测的情况，并且能够进行快速的填充。

3 算法效率分析

3.1 计算面积方法分析

两种算法计算面积的方法不同：种子填充算法通过获得一个点，然后递归调用搜索相邻的点，最后通过计数得到面积；扫描线算法则是通过一行一行的扫描，计算每一行的连续黑色像素点的面积，然后通过计算相互连通的行的面积来得到区域的面积。

对计算面积来说，扫描线算法的实现效率是比种子填充算法高。种子填充算法得到了面积后，填充时还需要使用递归来进行图像的填充操作，而扫描线算法则通过记录每行黑色区域的起止点即可进行填充。

3.2 遍历图片效率分析

两种算法都避免重复遍历图片。种子填充算法在遍历过的黑色像素点预先设定一个标识，而再次遇到该像素点时先判断标识，这样做就是要减少递归的次数，遇到检测过的像素点就不再递归。而扫描线算法是一行一行地进行像素点的扫描判断，所以并不存在重复遍历的情况，而扫描完所有的行数及遍历完整张图片。

比较两种算法的效率可以发现，递归的方法是无法控制递归深度的，只能尽量减少递归深度。目前种子填充算法可以通过链表存储来转换递归算法，但是其实现的效率并没有得到改善，所以在遍历图片上扫描线算法要比种子填充算法的效率高很多。在扫描大图像时，扫描线算法的速度更有优势。

3.3 填充图片效率分析

通过对种子填充算法的分析可以得出，种子填充算法在部分像素点上采用两次填充的方法，对黑色的像素点采用先设置标识然后填充的方法，以及二次填充。如果当黑色的像素点很多或为全黑的图像时，填充的效率为黑色像素数量的2倍。扫描线算法则根据记录需要填充的位置，采用一次填充的方法进行黑色像素点的判断和填充。因此，在对像素填充方面，扫描线算法上相对于种子填充算法的效率要高。

通过对两种算法的分析比较可以发现，扫描线算法的效率比种子填充算法的效率要高很多，种子填充算法在实现上的一个弊端是使用递归，虽然递归可以通过链表进行转换，但是实现的效率不高。而采用扫描线算法可以得到区域的形状，可以进一步扩充进行图像识别。对全黑图片的分析种子填充算法实现的效率为10万像素点/秒，而扫描线算法的实现效率为大约100万像素点/秒。因此可以说，使用扫描线算法比较合适。目前，对于区域填充扫描线算法的效率最高。

4 结束语

本文主要讨论二值图像噪声控制的两种算法实现方法，而对于灰度图像，则需要给定噪声的灰度范围，然后对噪声进行消除或平滑处理，只需要在算法上进行扩展即可实现。下一步我们将结合图像的识别，对不符合的图像进行处理，因此需要对相关内容进行研究。

参考文献：

1 唐泽圣，周嘉玉，李新有. 计算机图形学基础，北京：清华大学出版社，1995.

2 Donald Hearn, M Pauline Baker. Computer Graphics: C Version, Second Edition. Prentice Hall, 1996.

3 James D Foley, Andries van Dam, Steven K Feiner, et al. Computer Graphics: Principles and Practice in C (2nd Edition). Addison-Wesley Professional, 1995.

4 苗雪兰，刘瑞新. 计算机图形学理论及应用技术，北京：机械工业出版社，2003.

（责任编辑：黎贞崇）