参赛队号 # 1069

第七届“认证杯”数学中国

数学建模网络挑战赛
承 诺 书

我们仔细阅读了第七届“认证杯”数学中国数学建模网络挑战赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们接受相应处理结果。

我们允许数学中国网站(www.madio.net)公布论文，以供网友之间学习交流，数学中国网站以非商业目的的论文交流不需要提前取得我们的同意。

我们的参赛队号为：1069

参赛队员 (签名) ：

队员 1：洪光昊

队员 2：王洪飞

队员 3：黄怿晟

参赛队教练员 (签名)：

参赛队伍组别：本科组

参赛队号 # 1069

第七届“认证杯”数学中国

数学建模网络挑战赛

编 号 专 用 页

参赛队伍的参赛队号：（请各个参赛队提前填写好）：

1069

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

参赛队号 # 1069

2014年第七届“认证杯”数学中国数学建模网络挑战赛第一阶段论文

题　　目　位图的处理算法

关 键 词　Ostu+Canny算法 链码 抛物样条拟合 改进扫描线填充 GUI

摘　　要：

本次的问题是让我们处理栅格化逆过程，即根据位图图像像素点阵，得到矢量图形的几何特征，矢量图形本质上是使用曲线方程对图像进行精确描述，利用计算机辅助我们提取位图像素点阵特征，根据具体的拟合算法输出矢量图形。

首先，我们仔细分析图形信息，对图像进行了灰度处理，即对图像去彩色化。通过观察图像的灰度直方图，我们发现灰度直方图有双峰特性，采用ostu算法对图像进行了分割以及区域二值化，我们尝试了多种边缘检测算子，通过画图比较决定使用ostu+canny算子的方式进行边缘提取。从而得到边缘数据点。

然后，观察边缘数据点的分布情况，我们发现其在电脑内的存储状态没有构成点序列，出于对后面数据操作的便利，我们采用了freeman码的方式，对边界数据进行边缘跟踪，产生数据有序化，并在此过程中找出边界数据的突变点（尖点），为下一步的轮廓曲线拟合做准备。

接着，我们对有序化的边界数据进行了两种方式的拟合。第一种是采用抛物线样条曲线拟合，在之前的步骤中，我们得到了图形边界的尖点信息，由于整个图形是封闭的，我们以这些尖点作为分段点，在分段点区间内进行抛物线样条拟合，拟合标准便是所有点的偏差最小。从而得到边界轮廓方程。采取的第二种方法是针对于题目所给图像的特征，我们采取了分段函数拟合的办法，第一部分采用椭圆拟合的方式；第二部分是图形下部尖端部分，我们采用多项式拟合方式，最终得到边界图与边界方程。第一种方法具有一定的普适性，可以应用于任意形状的图形的拟合，拟合效果较为理想；第二种方法具有特殊性，针对有特定图像的拟合效果相对更好。

随后，在得到边界拟合曲线的基础上，我们综合比较了现在较为主流的种子填充算法和扫描线填充算法，并根据此次问题实际情况，提出了改进版本的扫描线填充算法，在进行边界内部填充的过程中，我们针对难填充的区域进行了区域划分，使得每一个区域都能够完美填充。

最后，我们考虑到图形程序较为复杂，与人的交互性较差，我们采用了简单的matlab gui界面方式对部分程序结果进行了统一展示，提高了程序的可视化以及人机交互性。

参赛队号： 1069

所选题目： B **题**

参赛密码 _____

（由组委会填写）

参赛队号 # 1069

英文摘要（选填）

（此摘要非论文必须部分，选填可加分，加分不超过论文总分的 5%）

Abstract

Rasterization is the task of taking an image described in a vector graphics format and converting it into a raster image. In this question, we study bitmaps into vector graphics.

Firstly, according to the bitmap, we take measures to generate Gray scale image and Binary image, we are aimed at the bitmap boundary which we use Integrated algorithm, Ostu algorithm and Canny operator, to extract boundary data.

Secondly, we observe the distribution of the boundary data. We find that we should make the boundary data well-aligned. So, we take two steps to solve this question. We use freeman codes to make data well-aligned so that we can extract the data which the slope has a big change.

Next, we take two measures to fit the data .the first method use Parabolic spline curve fitting, we can use the breakpoints as segmentation points and use the intervals to fit curve. Then, we can get equation of the boundary. Another method we used set piecewise equation to fit curve.The detail is the bottom of the image,which we solve with more patient.

Then, we have compared seed fill algorithm with scan line filling algorithm , and improved the scan line filling algorithm, which we can make area of filling needful.

Last but not least, we can improve the algorithm and design a interface to show our idea about this problem.

Key words：Ostu+Canny algorithm, Freeman Code, Parabolic spline curve fitting

参赛队号 # 1069

问题重述

图形图像在计算机里主要有两种存储和表示方法---位图和矢量图形[1][2]。位图是由像素点构成的图样，矢量图是根据几何特性来绘制图形，本质上是使用曲线方程对图像进行的精确描述。

位图的特点是可以精确反映图像细节，但是图像放大后会出现失真情况。矢量图靠软件记录几何特征生成，不能够反映图片细节，但是图形放大后不出现失真情况[3]。

将矢量图转换成以像素点阵来表示的信息，加以显示或打印的过程，称为栅格化过程。

本次问题研究的是栅格化的逆过程，即如何将一个由像素点阵组成的位图转换成矢量图形，通过建立合理的数学模型，恰当的方法来完成栅格化逆过程。

问题分析

这次的问题是栅格化的逆过程，也即是从位图到矢量化图形的过程。由于位图计算机存储的是每个像素点的信息，而矢量图形计算机存储的是其几何特征。所以在栅格化逆过程中我们必须要考虑位图图像的几何特征，并想办法提取这些几何特征。对于一个封闭图形，我们如果能够知道图形的边界方程、线条的粗细以及颜色，那么我们最后的工作就是展现这些特征所标识的矢量图形。

图形的边界方程的获取一般会涉及边界区域的确定，进而得到边界数据点，并将数据点有序化。一般对于边界的拟合有多边形近似、样条插值拟合和分段函数拟合，而大部分拟合方法要求数据的有序化。

线条的粗细可以根据边界提取后的边界数据来确定，而颜色可以通过对于位图像素点的提取来确定。

模型假设

假设 1：照片像素在 MATLAB 的处理范围内。
假设 2：填充的图形是封闭的，如果边缘本身有断点，用函数 bwfill 进行“补洞”操作；
假设 3：对于图片的比较微细小的部分，用膨胀函数 dilate 进行原图膨胀。

符号约定

$\sigma^2(k)$----阈值为 k 的类间方差；
$\omega(k)$----像素点出现的频率；
q----$|h^2 \times k|$表征曲率绝对值的大小；

参赛队号 # 1069

模型建立与结果展示

1 灰度图像

灰度使用黑色调表示物体,即用黑色为基准色，不同的饱和度的黑色来显示图像。每个灰度对象都具有从 0%（白色）到 灰度条 100%（黑色）的亮度值。 使用黑白或灰度扫描仪生成的图像通常以灰度显示。自然界中的大部分物体平均灰度为 18%。在物体的边缘呈现灰度的不连续性，图像分割就是基于这个原理。

1.1 图像的二值化

图像的二值化处理就是将图像上的点的灰度置为 0 或 255，也就是讲整个图像呈现出明显的黑白效果。即将 256 个亮度等级的灰度图像通过适当的阀值选取而获得仍然可以反映图像整体和局部特征的二值化图像。

2 图像分割

图像分割的目的在于将图像中人们感兴趣的部分从背景中分离出来，以进行后续的处理。分割的程度取决于要解决的问题，就是说在应用中当感兴趣的对象已经被分离出来时，就停止分割。因此，可以首先通过阈值处理将目标分离出来，并进行二值化，然后进行边缘跟踪得到目标的外边缘。

2.1 阈值处理

图像分割算法一般是基于亮度值的两个基本特征之一：不连续性和相似性。前者的应途径是基于亮度的不连续变化分割图像，比如图像的边缘。后者的主要用途是依据事先制定的准则将图像分割为相似的区域。阈值处理就是这一类。

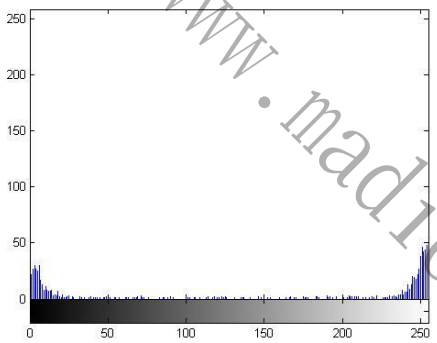


图 1.图片灰色直方图

由于阈值处理直观，实现简单且计算速度快，因此图像阈值处理在图像分割应用中处于核心地位。其基本原理如下：假设一幅图像有暗的对象和亮的背景组成，这样的组成方式将对象和背景具有灰度级的像素分成两组不同的支配模式。从背景中提取对象的一种显然的方法是选择一个阈值 T,将这些模式分离开，然后有$\mathbf{f(x,\ y) < T}$的点成为对象点：否则，就成为背景点。

对此我们采用 Ostu 方法的最佳全局阈值处理，该方法在类间方差最大的情况下是最佳的，即众所周知的统计鉴别分析中所用的度量。

设图像灰度级$\mathbf{1\sim M}$，第i级像素n_i个，总像素：

$$\mathbf{N} = \sum_{I=1}^{M} n_i$$

则第i级灰度出现的概率为

$$P_I = \frac{n_I}{N}$$

设灰度阈值为k，则图像像素按灰度级被分为两类：

$$C_0 = \{1,\ 2,\ \ldots,\ \mathrm{k}\},\ C_1 = \{K+1,\ \ldots,\ \mathrm{M}\}$$

图像的总平均灰度级：

$$\mu = \sum_{i=1}^{M} i \times P_i$$

C_0类的平均灰度级为：$\mu(k) = \sum_{i=1}^{M} i \times P_i$，像素数为：$\mathrm{N} = \sum_{I=1}^{k} n_i$

C_1类的平均灰度级为：$\mu - \mu(k)$，像素数为：$\mathrm{N} - N_0$

两部分图像所占比例分别为：

$$\omega_0 = \sum_{i=1}^{M} P_i = \omega(k)$$

$$\omega_1 = 1 - \omega(k)$$

对C_0，C_1均值做处理：

$$\mu_0 = \mu(k) / \omega(k)$$

$$\mu_1 = [\mu - \mu(k)] / [1 - \omega(k)]$$

图像总均值化为：$\mu = \omega_0\mu_0 + \omega_1\mu_1$

类间方差：

$$\sigma^2(\mathrm{k}) = \omega_1(k)[\mu - \mu_0]^2 + \omega_2(t)[\mu - \mu_1]^2$$

化解为：

$$\sigma^2(\mathrm{k}) = \frac{[\mu \cdot \omega_1(k) - \mu(k)]^2}{\omega(k) \cdot [1 - \omega(k)]^2}$$

K 从1~M变化，使$\sigma^2(\mathrm{k})$最大的k^*极为所求之最佳阈值。

$\sigma^2(\mathrm{k})$称为目标选择函数。

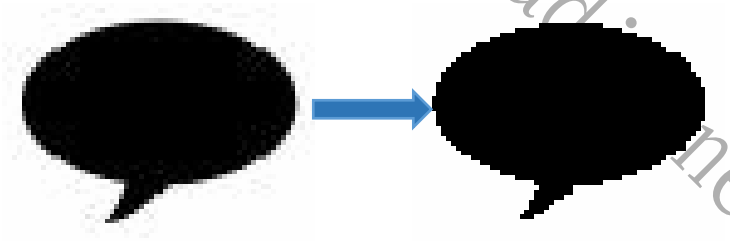


图 2 原图像　　图 3 Ostu 处理后图像

3 边缘检测

目前，边缘检测[4]最通用的方法是检测亮度的不连续。这样的不连续是用一阶和二阶导数来检测的，图像处理选择一阶导数为梯度。

2D 函数 $f(x,y)$ 梯度定义为如下矢量：

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

这个矢量大小是

$$\nabla f = mag(\nabla f) = \left[g_x^2 + g_y^2\right]^{\frac{1}{2}}$$

$$= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{\frac{1}{2}}$$

通常，这个数量用绝对值来近似

$$\nabla f \approx |g_x| + |g_y|$$

这个近似值避免了平方和开方运算，但是仍然具有微分特性（例如在常数区域 0，在像素值发生变化的区域，幅度与变化程度成比例）。在通常的应用中，把梯度的幅值简单的作为梯度。

梯度向量的基本性质是：梯度向量指向 (x,y) 坐标处 f 的最大变化率方向。最大变化率处的角度是

$$\alpha(x,y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

图像处理中的二阶导数通常用拉普拉斯方法来计算：

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

拉普拉斯自身很少被直接用作边缘检测，因为作为二阶导数，拉普拉斯对噪声的敏感性无法接受，他的幅度会产生双边缘，而且无法检测边缘方向。然而，当拉普拉斯方法与其他边缘检测技术组合使用时，拉普拉斯是强有力的补充方法。例如，虽然双边缘不适合直接坐边缘检测，但是这个`1 性质可通过寻找双边缘间的零交叉来边缘定位。

边缘检测的基本概念是用以下两个基本准则之一，在图像中寻找亮度发生快速变化的位置：

(1)寻找亮度的一阶导数的幅度比指定阈值大的地方。

(2)寻找亮度的二阶导数中有零交叉的位置。

3.1 Canny 边缘检测算子[5]

Canny 的目标是找到一个最优的边缘检测算法，最优边缘检测的含义是：

好的检测-算法能够尽可能多地标识出图像的实际边缘。

好的定位-标识出的边缘要与实际图像中的实际边缘尽可能接近。

最小响应-图像中的边缘只能标识一次，并且可能存在的图像噪声不应标识为边缘

3.2 Canny 算法的步骤

（1）降噪

任何边缘检测算法都不可能在未经处理的原始数据上很好地处理，所以第一步是对原始数据与高斯平滑模版作卷积，得到的图像与原始图像相比有些轻微的模糊。这样，单独的一个像素噪声在经过高斯平滑的图像上变得几乎没有影响。

（2）寻找图像的亮度梯度

图像中的边缘[6][7]可能会指向不同的方向，所以 Canny 算法[8]使用 4 个 mask 检测水平、垂直以及对角线方向的边缘。原始图像与每个 mask 所作的卷积都存储起来。对于每个店我们都标识在这个点上的最大值以及生成边缘的方向。这样我们就从原始图像生成了图像中的每个点亮度梯度图以及亮度梯度方向。

在图像中跟踪边缘

较高的亮度梯度比较有可能是边缘，但是没有一个确切的值来限定多大的亮度梯度是边缘，所以 Canny 使用了滞后阈值

滞后阈值需要两个阈值——高阈值与低阈值。假设图像中的重要边缘都是连续的曲线，这样我们就可以跟踪给定曲线中模糊的部分，并且避免将没有组成曲线的噪声像素当成边缘。所以我们从一个较大的阈值开始，这将标识出我们比较确信的真实边缘，使用前面导出的方向信息，我们从这些真正的边缘开始在图像中跟踪整个的边缘。在跟踪的时候，我们使用一个较小的阈值，这样就可以跟踪曲线的模糊部分直到我们回到起点。一旦这个过程完成，我们就得到了一个二值图像，每点表示是否是一个边缘点。

一个获得亚像素精度边缘的改进实现是在梯度方向检测二阶方向导数的过零点

$$L_x^2\,L_{xx} + 2\,L_x\,L_y\,L_{xy} + L_y^2\,L_{yy} = 0,$$

它在梯度方向的三阶方向导数满足符号条件

$$L_x^3\,L_{xxx} + 3\,L_x^2\,L_y\,L_{xxy} + 3\,L_x\,L_y^2\,L_{xyy} + L_y^3\,L_{yyy} < 0$$

其中L_x, L_y ... L_{yyy}表示用高斯核平滑原始图像得到的尺度空间表示L计算得到的偏导数。用这种方法得到的边缘片断是连续曲线，这样就不需要另外的边缘跟踪改进。滞后阈值也可以用于亚像素边缘检测。如下图所示：

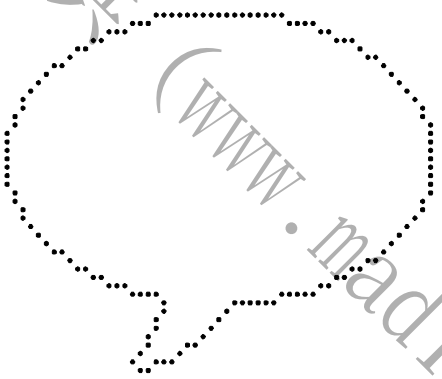


图 4 Ostu+Canny 的边缘检测

4 边缘跟踪

在对目标进行分离之后，就可以采用边缘跟踪。我们采用 8 方向 freeman 链码对图像中的目标进行边缘跟踪。链码用于表示由顺次连接的具有制定长度和方向的直线段组成的边界，典型的，这种表示基于这些线段的 4 连接或 8 连接。边界的链码取决于起始点的绝对坐标与各个偏移量。

4.1 像素的连接性

对一个坐标为$\mathbf{p}(\mathbf{x},\mathbf{y})$的像素点，它可以有四个水平和垂直的临近像素，它们的坐标分别为$(\mathbf{x}+\mathbf{1},\mathbf{y})$，$(\mathbf{x}-\mathbf{1},\mathbf{y})$，$(\mathbf{x},\mathbf{y}+\mathbf{1})$，$(\mathbf{x},\mathbf{y}-\mathbf{1})$。这些像素点组成$\mathbf{p}$的 4-邻域，记为$N_4(p)$。坐标$(\mathbf{x},\mathbf{y})$的像素与各个它各个四邻域近邻像素是一个单位距离。像素点$\mathbf{p}(x,y)$的 4 个对角近邻像素的坐标是$(\mathbf{x}+\mathbf{1},\mathbf{y}+\mathbf{1})$，$(\mathbf{x}+\mathbf{1},\mathbf{y}-\mathbf{1})$，$(\mathbf{x}-\mathbf{1},\mathbf{y}+\mathbf{1})$，$(\mathbf{x}-\mathbf{1},\mathbf{y}-\mathbf{1})$，它们记为$N_D(p)$. 这些像素点加上 4-邻域像素合称为$\mathbf{p}$的 8-领域。

4.2 像素的连通性问题

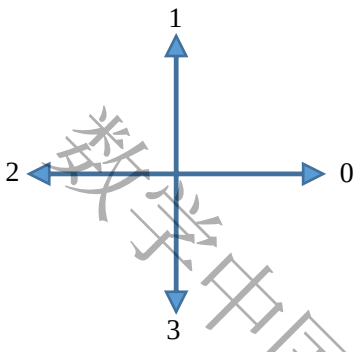
像素间的连通性是一个基本概念。它对许多光栅图像的定义做了简化，如区域和边

界。为定义像素间的连通性，我们做定义：

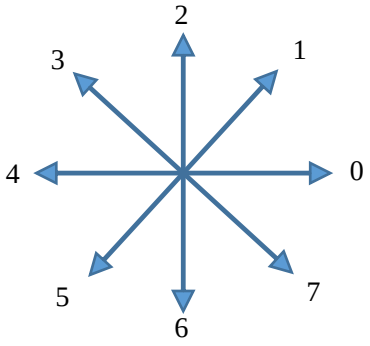
（1）4-邻接：2 个像素 p 和 r 在 V 中取值且 r 在 N4（p）中，则他们为 4-邻接；

（2）8-邻接：2 个像素 p 和 r 在 V 中取值且 r 在 N8（p）中，则他们为 4-邻接；

下图为 4-邻接，8-邻接方向示意（a），（b）：



图（a）



图（b）

对于一个八连通的图像区域来说，有以上定义可知，区域中的每个像素，它的周围总存在 8 个像素点与它连接，可以为这八个像素设定从 0 到 7 的方向编号，如图　所示如此一旦确定像素 P 的位置，以及某个邻域像素的编码就可以知道邻接像素的位置。链码实际上是一串指向符的序列。链码表示就是从某点开始观察某一曲线的走向并用相应的指向符来表示，结果形成一个序列，因此可以用链码来描述任一曲线或闭合的边界。

此文中用链码对图像中的目标进行跟踪的过程如下:对整个图像进行由上到下，由左到右的扫描，当发现一个作为前景颜色的像素（白点），则对和这一点相关的图形进行外边缘的跟踪。

对每个特定目标进行外边缘跟踪的过程如下：从第一个边界点开始，定义起始点的搜索方向为左上方；如果左上方的点是白点，则为边界点，否则搜索方向顺时针旋转 45 度。直到找到第一个白点未知这个白点就是新的边界线。新的边界点的初始值搜索方向为前一边界点到该点方向的反方向并顺时针旋转 45 度，用同样的方法搜索下一个点，直到返回最初的边界点为止。

本文对对图片中的目标进行链码跟踪，根据起始点的坐标及当前码值将链码转换成边缘的坐标点序列。

5 边缘曲线的分段拟合

5.1 基本概念

图像分割后，目标的边缘轮廓被表示成一个个分散的像素点，数据量非常大。因此，通常的处理方式是用一系列首位相连的曲线来拟合这些点序列，以达到最终矢量化的目的。轮廓曲线通常都采用内插曲线或者逼近曲线来实现拟合。

5.1.1 插值，逼近和拟合

(1)插值

插值是函数逼近的重要方法。给定函数$f(x,y)$在区间$[a,\ b]$中互易的n个点的值$f(x_i)$，$i=1,\ 2,\ \ldots n$。基于列表数据，寻找某一函数$\varphi(x)$去逼近$f(x)$。若要求逼近函数$\varphi(x)$在x_i与$f(x_i)$相等，就称这样的函数逼近问题为插值问题，称$\varphi(x)$为$f(x)$的插值函数。

也就是说，$\varphi(x)$在 n 个插值节点x_i处与$f(x_i)$相等，而在别处就用$\varphi(x)$近似代替$f(x)$。在曲线中最常用的是线性插值和抛物样条插值。

(2)逼近

当型值点太时，构造插值函数使其通过所有型值点相当困难。这时，往往选择一个次数较低的函数，在某种意义上最佳逼近这些型值点。逼近常采用最小二乘法。假设已知一组型值点(x_i, y_i)，$i = 1, 2, \ldots n$。要求构造一个$m(m < n-1)$次多项式$y = F(x)$逼近这些型值点。逼近这些型值点。逼近的好坏主要看是否使个点的偏差的平方和最小，即;

$$\varphi = \sum_{k=1}^{n} [F(x_k) - y_k]^2$$
$$(k = 1,2,\ldots n)$$

上式值较小。

(3)拟合

拟合并没有像上述的插值，逼近那样有完整的定义和数学表达。拟合是指在曲线，曲面的设计过程中，用插值或者逼近的方法生成的曲线，曲面达到某些设计要求，例如在允许的范围内贴近原始的型值点或控制点序列等，在本篇文章中，我们采用抛物样条插值的方法拟合原始的点序列，目的是使目标边缘更加光滑自然。

5.1.2 算法评价标准

评价一个拟合算法效果的好坏的标准主要是以下两个方面：

（1）精确性：拟合后的曲线段应与原始的边缘点序列误差比较小，并能表示处原始轮廓的特征，如在本系统中要求拟合曲线与原始点序列的最大误差不能过大。

（2）数据压缩率：曲线拟合应当提高轮廓表示的经济性，能为图像的后处理提供一种更简单，更紧凑的表示。

事实上曲线拟合的高效性及精确性主要有待拟合曲线的特点和拟合算法本身的性能决定。综合这些特点本文利用两种不同角度的拟合方法对图样的边缘进行矢量化。这种方法如下：

Step1：尖点提取，即通过曲率方法提取出轮廓中的尖点，防止在下一步的拟合中将尖点平滑掉，曲线特征丢失。

Step2：曲线拟合，即以尖点为分段点，对每两个尖点之间的曲线采用曲线进行拟合，直到精度满足一定条件为止，以下将会详细介绍。

5.2 尖点提取

我们知道尖点是对应曲线中具有局部曲率极大值的点，即曲线的斜率变化发生突变的点。，但在拟合时无法用光滑的曲线表示尖点，因此通常先提取处边缘中的尖点，然后把两个尖点之间的曲线作为一个拟和区域。

通常我们取尖点大都利用曲率局部极大值的方法，但由于在数字曲线中，无法直接用公式准确的计算出某一点的曲率值，所以只能计算某种近似值。由于直接利用曲线上点的坐标来计算曲率值，比较复杂，本文采用一种间接的算法，该算法相对其他的算法计算量小，算法简单，并且适合任意形状的曲线，能快速准确的提取出轮廓上的尖点。

5.2.1 算法理论分析

经过边缘的有序化处理[9]，们知道第i个点即$P(i)$点的坐标由$X(i)$和 $Y(i)$共同决定，因此,我们可以将$P(i)$分解为两条以为离散变化曲线$X(i)$和 $Y(i)$,$X(i)$的变化体现了$P(i)$水平方向的变化过程， $Y(i)$则体现了$P(i)$在竖直方向的变化情况。所以只要我们分别找到

曲线X(i)上的尖点和曲线Y(i)上的尖点，综合起来，便能找到P(i)上尖点集合。由于X(i)和 Y(i)都是关于 i 的一元离散函数，所以曲率的估算变得简单明了。

5.2.2 算法的数学表达

在对X(i)和 Y(i)尖点的提取中，我们根据的是曲率绝对值的大小进行提取，因此并不需要知道曲率本身的准确值，只要根据一个数值判断其绝对值大小及对曲线上的点进行取舍即可。

以X(i)为例，在数值分析中，对离散的函数 X=X(i)，以$i=1,\ 2,\ \ldots n-1$为等距节点，由插值原理可得：

$$X^n(i) \approx \frac{1}{h^2}[X(i-h)-2X(i)+X(i+h)]$$

而曲线上点的二阶导数近似等于该点的曲率，则X(i)上第i点的曲率为k，则有：

$$k \approx \frac{1}{h^2}[X(i-h)-2X(i)+X(i+h)]$$

考虑到数字化和噪声的影响，我们在算法中区步长$h=10$，令$q=|h^2\times k|$

则有：

$$q \approx |X(i-h)-2X(i)+X(i+h)|$$

同理，Y(i)上的个点的 q 值计算方法与X(i)中一样。

5.2.3 尖点分析

以下几个图是用MATLAB语言实现该数学算法并对图中的边缘提取尖点得到的结果，图 5 是在曲线X(i)和Y(i)上提取的尖点情况，图 6 则是对两个方向上的尖点进行尖点综合得到的结果。

从结果可以看出该算法对尖点的定位是比较准确的，且适应各种形状。在算法效率上，本算法在两个方向上的曲率计算分别只用到数次加减法运算，与其他方法相比，计算量大大减少，效率高。

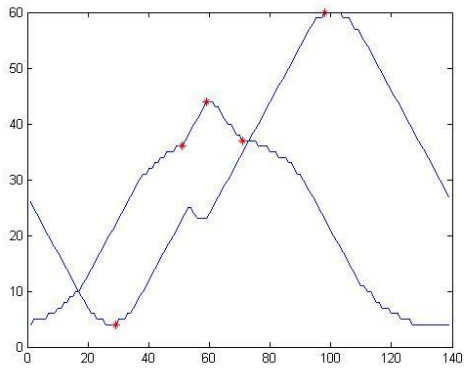


图 5 x(i),y(i)的尖点

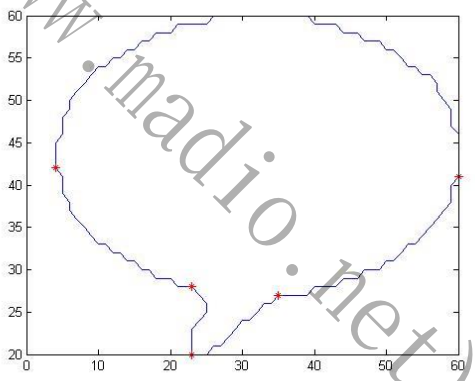


图 6 边缘点上的尖点

而前文提到的所谓间接方法实际上是一种基于参数方程的方法，在此方法中，我们通过引用一个新参数，也就是序列号 i，利用该参数，我们即将点序列P(i)的坐标转化成方X(i)，Y(i)的方法来表示

$$\begin{cases} X(i) = X_i \\ Y(i) = Y_i \end{cases}$$

其中X(i)，Y(i)是关于 i 的一维离散函数。相较于目前很多数字曲线的处理方法，都要求坐标y是坐标x的函数，如最小二乘拟合法。这种引入参数方程以后，我们可以把一条闭合曲线表示成两条离散函数曲线，因此就可以把这些方法应用到闭合曲线的处理上。因此在此提到的方法在很多大数字曲线的处理中也很有意义。

5.3 曲线的拟合

5.3.1 抛物样条曲线拟合

我们知道矢量中用来拟合的曲线通常包括线段，样条曲线，圆弧及椭圆弧等。在本系统中，由于边缘的线条具有多样性，而且考虑到算法的兼容性，本文采用形状比较灵活的抛物样条曲线作为拟合基元。而通过不在同一直线的三点：P_1, P_2, P_3，定义一条抛物样条曲线的 表达形式如下：

$$P(t) = A_1 + A_2 t + A_3 t^3$$
$$0 \le t \le 1$$

该抛物线样条曲线经过P_1, P_2, P_3三个点，并且有：

（1） 抛物线段以P_1为起始点。即当$t = 0$时，曲线过P_1点。

（2） 抛物线段以P_3为终点。即当$t = 1$时，曲线过P_3点。

（3） 当参变量$t = 0.5$时，曲线过P_2点，且切矢量等于$P_3 - P_1$。

由以上条件约束可得：

$$\begin{cases} A_1 = P_1 \\ A_1 + A_2 + A_3 = P_3 \\ A_1 + 0.5A_2 + 0.25A_3 = P_2 \end{cases}$$

解得：

$$\begin{cases} A_1 = P_1 \\ A_2 = 4P_2 - P_3 - 3P_1 \\ A_3 = 2P_1 + 2P_3 - 4P_2 \end{cases}$$

于是得到：

$$P(t) = (2t^2 - 3t + 1)P_1 + (4t - 4t^2)P_2 + (2t^2 - t)P_3$$
$$0 \le t \le 1$$

此拟合基元实际上是直线和抛物样条曲线。$P(t)$是一个点向量，在二维平面上他包含里两个坐标值$[X(t), Y(t)]$。以上推导求出的算式，即我们要求过不在一直线上的三点：$P_1(x_1, y_1)$，$P_2(x_2, y_2)$和$P_3(x_3, y_3)$的抛物线方程。对该条抛物线进行存储和编辑时，只需对P_1，P_2，和P_3进行操作。无需像数字曲线那样对每个点都进行存储和编辑。根据 t 的取值即可在屏幕上显示该曲线。

5.3.2 递归算法

为满足精度和压缩率的要求，我们通常使用遗传算法或递归算法，但由于遗传算法计算量大，结构复杂。所以，本文使用一种递归的拟合方法对各个尖点之间分别进行拟合。

该算法的思想是:以当前离散曲线段的起点，中点，终点构造一条抛物样条曲线对当前曲线段进行拟合并进行误差计算，如果本段拟合的误差小于给定的阈值，返回；否则以当前曲线段的中点将曲线分为两部分，在两个曲线段分别利用同样的方法进行拟合，一次类推，直至本段的误差小于给定的阈值时为止。

利用这种方法一次拟合各尖点之间的曲线段，便得到了总的拟合结果。可以看出该算法对每段曲线只需要几个简单地递归语句就能实现拟合，非常简单且易于实现。

5.3.3 拟合算法运行结果

运行以上拟合算法，利用 MATLAB 得到

$$P(t) = (2t^2 - 3t + 1)P_1 + (4t - 4t^2)P_2 + (2t^2 - t)P_3$$

中每段对应的系数如下表：

P	x_1	x_2	x_3	Y_1	Y_2	Y_3
P_1	4	6	9	26	19	12
P_2	9	15	22	12	6	4
P_3	22	24	27	4	5	6
P_4	27	30	32	6	9	12
P_5	32	34	36	12	17	23
P_6	36	40	44	23	24	23
P_7	44	41	37	23	29	35
P_8	37	36	34	35	41	48
P_9	34	30	23	48	55	60
P_{10}	23	18	13	60	60	57
P_{11}	13	10	7	57	53	48
P_{12}	7	4	4	48	38	27

表 1

拟合后的图像如下所示：

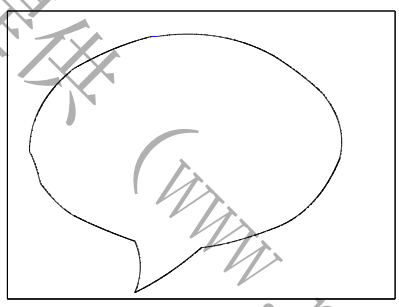


图 7 抛物样条曲线拟合

5.4 针对特定特征图标的轮廓拟合

5.4.1 图标特征分析

我们通过对给定的图标进行特征分析，观察图形轮廓是否是某几个特征元素组成，如：点、线、矩形、多边形、圆和弧线。我们可以从边缘检测的结果看出，该图标可以视为三部分组成：椭圆和尖端（两条弧线）。那么，在这种情况下，我们就可以简化曲线拟合的算法，不需要完全对尖点划分的多曲线段进行一次拟合，而可以将多个有椭圆方程性质的段一起进行椭圆方程拟合，从而减少了拟合的曲线段，使拟合后的曲线更加光滑，更能得到的拟合效果会更好些。但此时会引入一个问题：当我们减少了对边界段（尖点分段）的拟合，就可能会丢失一些图形的细节信息。在进行任意形状复杂的轮廓边界拟合时，其细节信息往往很重要，故此时的针对特定特征图标的轮廓拟合算法的效果不一定好。基于此图标的特征分析，我们将进行边缘曲线拟合，检验方程拟合优度，发现其效果较好一些。

5.4.2 部分边缘的椭圆拟合

（1）椭圆曲线拟合基本原理

椭圆曲线拟合对于给定平面上的一组样本点，寻找一个椭圆，使其尽可能靠近这些样本点。也就是说到，将图像中的一组数据以椭圆方程为模型进行拟合，使某一椭圆方程尽量满足这些数据，并求出该椭圆方程的各个参数。最后确定的最佳椭圆的中心即是搜们要确定的靶心。

椭圆曲线拟合最早的拟合方法是最小二乘法，它是数据拟合中的基本方法，它的基本思想就是考虑数据受随机噪声的影响进而追求整体误差的最小化。对椭圆拟合而言，就是先假设椭圆参数，得到每个待拟合点到该椭圆的距离之和，也就是点到假设椭圆的误差，求出使这个和最小的参数。我们使用的是非线性多元回归拟合，采用 regress 函数针对二元曲线拟合，得到拟合检验优度与残差图，进行拟合检验。

（2）椭圆拟合

我们利用链码找出椭圆与两条弧线的分界点（即其中的一个尖点）对类椭圆的部分边缘线段进行椭圆方程拟合，得到如下的图形：

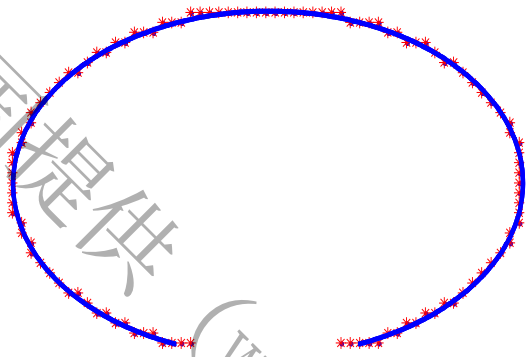


图 8 椭圆曲线拟合

从拟合的效果来看，有针对性的椭圆边缘拟合，与数据的拟合优度更高，更能反映原边缘的信息。显然，利用此拟合后的曲线，进行的填色处理得到的矢量图会更能反映原图的特征。

利用椭圆拟合算法获取的椭圆方程形式，如下：

$$F1(x,y)=-0.0019x^2-0.0007y^2+0.0786x+0.0468y-1$$

关于拟合得到的椭圆方程的拟合优度检验，在模型检验部分有详细叙述，这里不再赘述。

5.4.3 部分边缘的多项式拟合

（1） 多项式拟合基本原理

假设给定数据点 (x_i,y_i) (i=0, 1, …, m)，Φ 为所有次数不超过 $n(n\le m)$ 的多项式构成的函数类，现求一 $p_n(x)=\sum_{k=0}^{n}a_k x^k\in\Phi$，使得

$$I=\sum_{i=0}^{m}[p_n(x_i)-y_i]^2=\sum_{i=0}^{m}\left(\sum_{k=0}^{n}a_k x_i^k-y_i\right)^2=\min \qquad (1.0)$$

当拟合函数为多项式时，称为多项式拟合，满足式(1.0)的 $p_n(x)$ 称为最小二乘拟合

多项式。特别地，当 n=1 时，称为线性拟合或直线拟合。

显然

$$I=\sum_{i=0}^{m}(\sum_{k=0}^{n}a_k x_i^k-y_i)^2$$

为$a_0,a_1,\cdots a_n$的多元函数，因此上述问题即为求$I=I(a_0,a_1,\cdots a_n)$的极值 问题。由多元函数求极值的必要条件，得

$$\frac{\partial I}{\partial a_j}=2\sum_{i=0}^{m}(\sum_{k=0}^{n}a_k x_i^k-y_i)x_i^j=0,\qquad j=0,1,\cdots,n \tag{2.0}$$

即

$$\sum_{k=0}^{n}(\sum_{i=0}^{m}x_i^{j+k})a_k=\sum_{i=0}^{m}x_i^j y_i,\qquad j=0,1,\cdots,n \tag{3.0}$$

（3.0）是关于$a_0,a_1,\cdots a_n$的线性方程组，用矩阵表示为

$$\begin{bmatrix} m+1 & \sum_{i=0}^{m}x_i & \cdots & \sum_{i=0}^{m}x_i^n \\ \sum_{i=0}^{m}x_i & \sum_{i=0}^{m}x_i^2 & \cdots & \sum_{i=0}^{m}x_i^{n+1} \\ \vdots & \vdots & & \vdots \\ \sum_{i=0}^{m}x_i^n & \sum_{i=0}^{m}x_i^{n+1} & \cdots & \sum_{i=0}^{m}x_i^{2n} \end{bmatrix}\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}=\begin{bmatrix} \sum_{i=0}^{m}y_i \\ \sum_{i=0}^{m}x_i y_i \\ \vdots \\ \sum_{i=0}^{m}x_i^n y_i \end{bmatrix} \tag{4.0}$$

式（3.0）或式（4.0）称为正规方程组或法方程组。

可以证明，方程组（4.0）的系数矩阵是一个对称正定矩阵，故存在唯一解。从式（4.0）中解出a_k (k=0, 1, …, n)，从而可得多项式

$$p_n(x)=\sum_{k=0}^{n}a_k x^k \tag{5.0}$$

可以证明，式（5.0）中的$p_n(x)$满足式（1.0），即$p_n(x)$为所求的拟合多项式。我们把$\sum_{i=0}^{m}[p_n(x_i)-y_i]^2$称为最小二乘拟合多项式$p_n(x)$的平方误差，记作

$$\|r\|_2^2=\sum_{i=0}^{m}[p_n(x_i)-y_i]^2$$

由式(2.0)可得

$$\|r\|_2^2=\sum_{i=0}^{m}y_i^2-\sum_{k=0}^{n}a_k(\sum_{i=0}^{m}x_i^k y_i) \tag{6.0}$$

多项式拟合的一般方法可归纳为以下几步：

Step1：由已知数据画出函数粗略的散点图，确定拟合多项式的次数 n；

Step2： 列表计算

$$\sum_{i=0}^{m}x_i^j\qquad(j=0,1,\cdots,2n)$$

$$\sum_{i=0}^{m}x_i^j y_i\qquad(j=0,1,\cdots,2n)$$

；

Step3：写出正规方程组，求出$a_0, a_1, \cdots a_n$；

Step4：写出拟合多项式

$$p_n(x) = \sum_{k=0}^{n} a_k x^k$$

。

在实际应用中，$n < m$或$n \le m$；当$n = m$时所得的拟合多项式就是拉格朗日或牛顿插值多项式。

(2) 多项式拟合

利用的链码找出尖点，确定尖端的数据位置，将尖端分成两左右部分弧线拟合，针对这两部弧线的拟合，我们就可以使用多项式进行拟合。

左边尖端弧线的多项式拟合，进经过 4 次多项式拟合，拟合结果如下图所示：

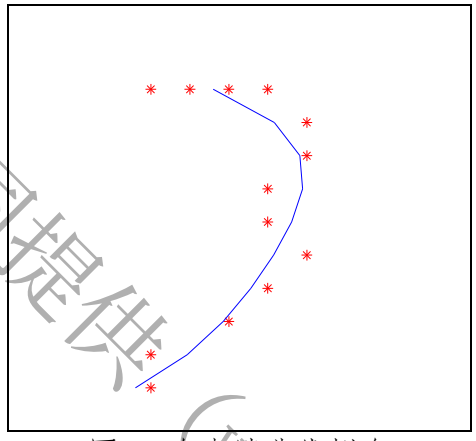


图 9 左尖端曲线拟合

拟合的曲线方程如下所示：

$$F2(x,y) = -0.0035x^4 + 0.5976x^3 - 38.0287x^2 + 1076.4814x - 11407.1918$$

左边尖端弧线的多项式拟合，经过 5 次多项式拟合，拟合结果如下图所示：

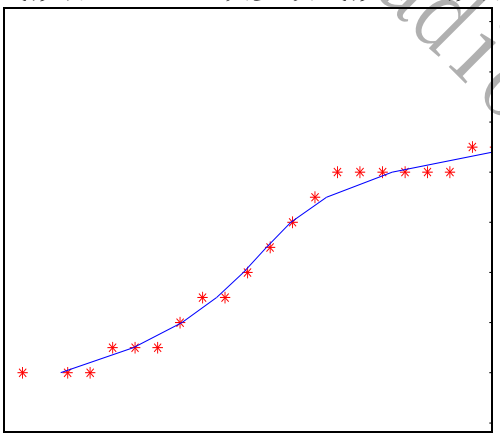


图 10 右尖端曲线拟合

拟合的曲线方程如下所示：

$$F3(x,y) = -0.0013x^5 + 0.2658x^4 - 22.5839x^3 - 22.5839x^2 - 20433.1829x + 174104.9984$$

关于拟合得到的左边尖端弧线方程的拟合优度检验，在模型检验部分也有详细叙述，这里不再赘述。

5.5 针对特定特征图标的轮廓拟合

考虑到椭圆与尖端曲线的拟合，结合各个部分的方程，我们可以做出整个轮廓的边界轮廓，即矢量化，图形如下：

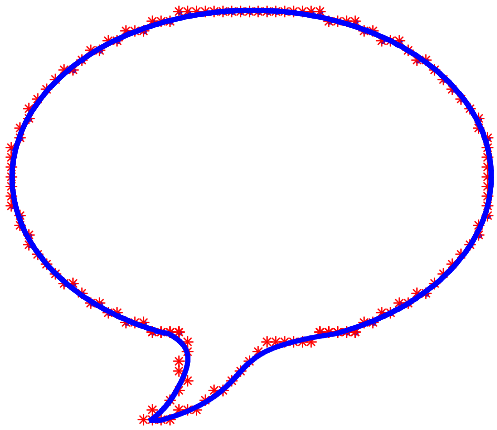


图 11 特定特征图标的轮廓拟合

至于各个部分的拟合很好，综合各个段的拟合拟合方程，我们得到整体拟合效果也能很好的反映原图形的特征，显然，上图也可以很好的反映拟合效果。我们可以看出：这种有针对性的特定特征图标的轮廓拟合，相比于任意曲线的尖点分段拟合的效果很好，这种算法有一定的局限性，须在给定的图标是几个元素组成的（点、线、矩形、多边形、圆和弧线）。

6 轮廓图形的填充[10][11]

在上面的讨论中我们研究了一个位图的边缘检测以及边界点提取，并讨论了边界点数据的方程拟合，当我们拟合出边界方程后，需要完整的将栅格化逆过程（即将位图矢量化）完成，我们还需要将上述提取的边界轮廓填充起来，使其在颜色上与原位图保持一致，我们查阅文献，发现现有的主流填充算法有扫描线算法，种子填充算法[12][13]等。

种子填充法又称边界填充法。其基本思想是：现将区域的一点赋予指定的颜色，然后将该颜色扩展到整个区域的过程。种子填充要求区域是连通的。

种子填充的算法常用的是四向连通域和八向连通域的填充。四向连通算法：从区域内任意一点出发，通过上、下、左、右四个方向到达区域内的任意像素；而八向连通算法：从区域内任意一点出发，通过上、下、左、右、左上、左下、右上和右下八个方向到达区域内的任意像素。在算法实现时它们的区别在于是仅通过上、下、左、右 4 个方向，还是从左下、左、左上、上、右、右上、右下、下 8 个方向寻找下一个象素。

这两种的填充具体操作实现相比于其他填充的算法容易，且可以用于填充带有空洞的平面区域。以四向连通填充算法为例，其具体实施如下：

（1） 将种子像素压入栈中；

（2） 进行判断：如果栈中为空，则转至连通域和内点；否则转至步骤（3）；

（3） 弹出一个像素，并将该像素置成填充色；并判断该像素相邻的四连通像素是否为边界色或已经置成多边形的填充色，若不是，则将该像素压入栈；

（4） 循环执行以上步骤，即转至过程（2）；

（5） 结束。

这种算法使用了递归思想，给实施带来了便利，但其暴露了一个缺点：算法需要很大的存储空间以实施栈结构，这是由于当某一像素出栈时，都会将周围的 4 个像素进行入栈，不论其是否被填充。就会产生同一个像素入栈多次的现象。

基于以上种子填充法的可能存在的不足，考虑到栅格化的逆过程处理的一般是一些较为简单的图标，兼顾算法的计算机实施的简易性与填充准确性，在这里我们针对此问题采用的是改进化的扫描线填充算法。

经典扫描线算法的基本思想是：首先利用区域在扫描线上的连贯性，填充当前扫描线上的一个区段，然后利用相邻扫描线上区段的连贯性，在上下两条相邻扫描线上寻找新的区段，并以种子点堆栈的形式保存这些区段. 反复处理堆栈内的区段，直到保存的所有区段都处理完为止. 这一过程可以分为如下 5 个步骤进行：

（1）初始化：将算法设置的堆栈置为空，将给定的种子点（x，y）入栈；

（2）出栈：如果堆栈为空，算法结束；否则取栈顶元素（x，y）作种子点；

（3）区域填：从种子点（x，y）开始沿纵坐标 y 的当前扫描线向左右两个方向逐个像素进行填色，其值 CFilling，直到到达为止；

（4）确定范围：以$[x_l,x_r]$和x_r分别表示在步骤 3 中填充的区段两端点的横坐标；

（5）入栈：分别确定当前扫描线的上下相邻两条的扫描线位于区域内的区段。如果这些区段的像素的对应的颜色值为填充色 CFilling 或者为轮廓的颜色 CBoundary，则不产生新的区段，直接转到步骤 2 ，否则取区段的右端点为种子点入栈，再转到步骤 2 继续执行。

我们可以发现，在上述算法执行的过程中，我们重点需对一个像素点进行判断，观察其是否为区域的内点，同时填充的时候，种子点开始向左右两个方向进行像素点的读取与判断，这种重复的操作在一定程度上影响了算法实施的效率，为此我们提出了使用改进化的扫描线填充算法，其有以下改进：

A. 针对轮廓较为简单的图标，我们尽量避开对填充域内点即种子点的选取操作，改为图片边缘开始扫描，扫描方向可以是图片的任意一个边界，依次扫描，完成填充；

B. 针对图形的轮廓较为复杂的图标，我们还相应设置了双向扫描填色的算法，在填充的过程中，我们可能会遇到个别区域填充不完全或未填充的现象，我们采用了分割图像的方法，将不好填充的区域划分成一些好填充的区域，分布填涂，这样提高了填色的效率和准确性。

我们通过之前拟合方程的过程，得到如图 12 所示的轮廓线图形。

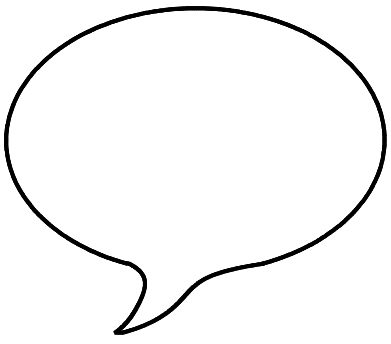


图 12 轮廓线

我们根据这个图形的结构，选择从左往右开始逐行开始扫描该二值化的图形，依据的原理概要如下：

（1）如果像素值为 1（在此处计算机显示为白色，即背景色），不做改变继续向右

边扫描；

（2）直到遇到像素点为为 0（黑色），此时这个点的下一个像素值若为 1，则表明计算机扫描到了图形的最左边的轮廓点，这时候设置一个标记 Flag1，记录该位置的信息；

（3）从这点往后所有像素点值为 1 的全部置为 0（填充黑色），直到再次遇到像素值为 0 的点为止，即扫描到轮廓的一个右边界，设置另一个标志 Flag2；

（4）跳转至步骤（1），同时进行标志位 Flag2 与像素值为 1 的判断，若满足该条件，则表明再次扫描到后面的轮廓，须继续以上的过程；若不满足该条件，则表明已经扫描到所有的轮廓线，则跳转至步骤（5）；

（5）已经扫面到存在的轮廓线，并完成填色，算法结束。

以图片的左边界对应的各行为起始位置 x_s 扫描起点，进行正行图片的扫描，如下图 13 所示

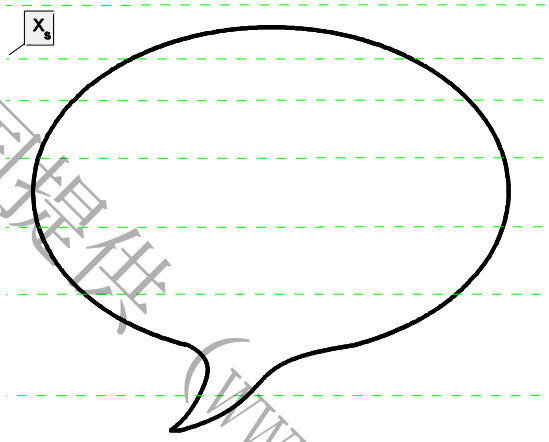


图 13 改进后的扫描线填充

通过之前所述步骤，我们可以完成对所给的封闭图形的填充，得到填充后的图形，如下图 14 所示：

图 14 填充后抛物样条拟合和针对特定特征图标的轮廓拟合

此时，我们便得到了矢量图 3，完成了栅格化的图形的逆过程，其边界关系是一个椭圆函数和多项式函数组成，这些图形的元素是一些点、线、矩形、多边形、圆和弧线等组成，用了可以放大、缩小或者旋转，均不失真的性质。

模型检验

(1) 图像边缘提取比较

在对矢量图形进行边缘提取时，我们利用 matlab 软件比较了几种边缘检测方法的好坏，比较结果如下图：

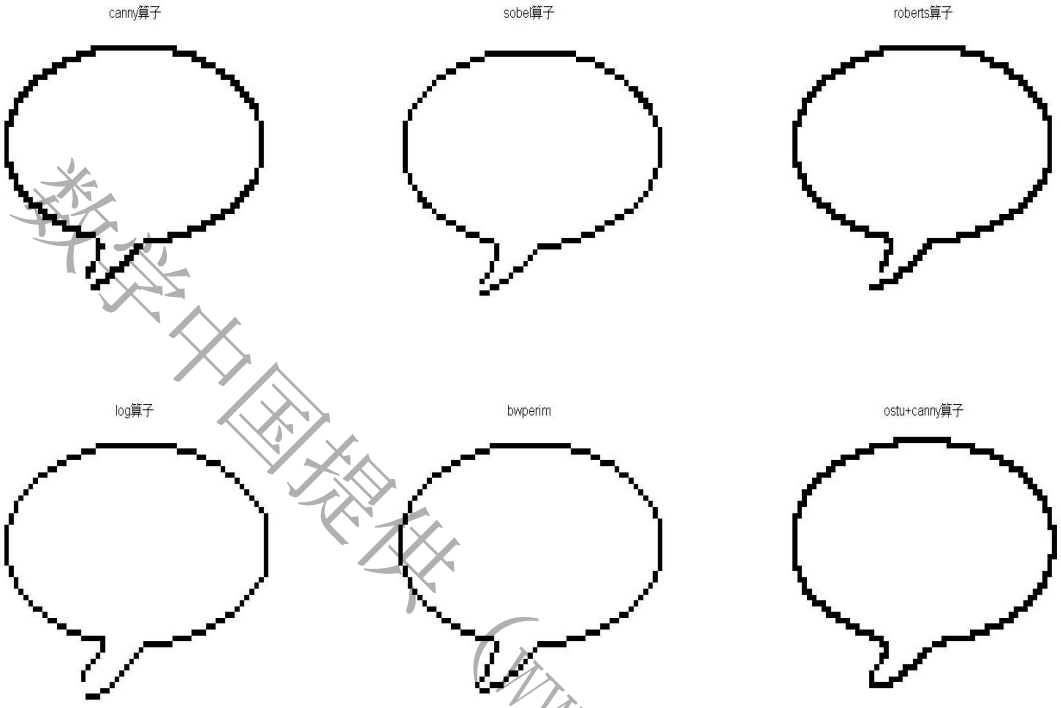


图 15 不同算子效果比较

通过比较 Canny 算子、Sobel 算子、Roberts 算子、Log 算子、Matlab 提供的二值图像边缘提取函数 bwperim 以及我们自己考虑的综合 ostu 聚类算法和 canny 算子产生的边缘提取图像，我们发现综合算法使得图像提取更加完整，其余的方法提取的图像边缘存在断点的情况，效果并不理想。

(2) 椭圆拟合与尖端曲线拟合

拟合指标：P=0.0002； R^2 =0.99987；

由此可知椭圆拟合效果较好

拟合的残差图形：

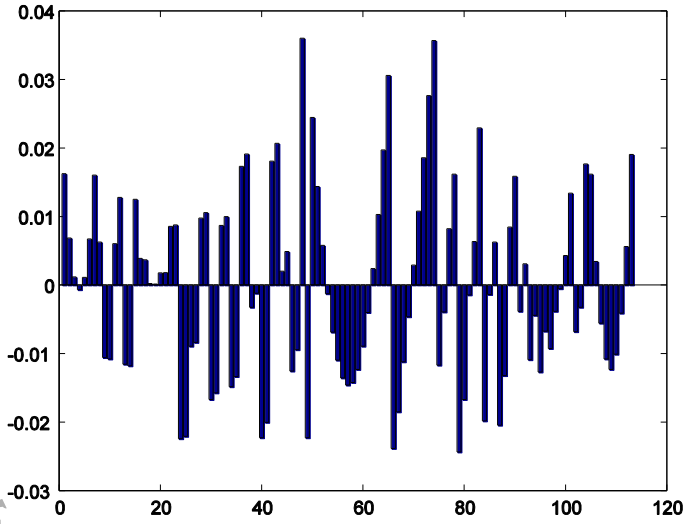


图 16 椭圆拟合残差图

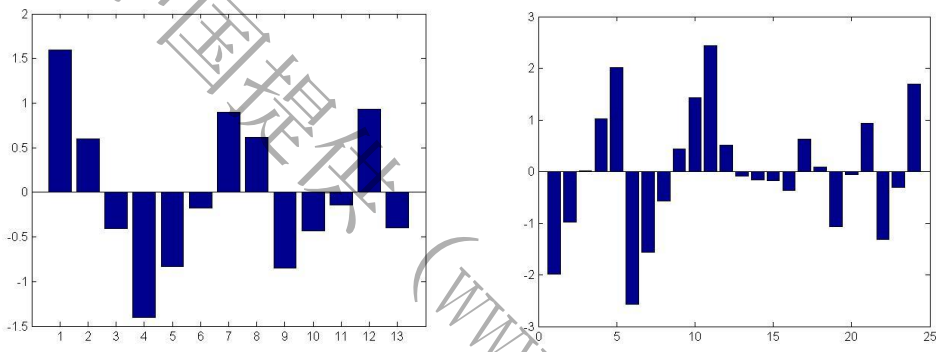


图 17 尖端拟合左端，右端残差分析

模型评价

这次的模型较好的解决了位图转换成矢量图形[14]这一问题。其中我们使用了Ostu+Canny 算子的组合方式提取位图边缘轮廓，freeman 码方式使边界数据有序化，并且使用了多种拟合方式，并对多种拟合方式进行了比较。这点充分展示了数学模型的特点，在填色步骤中，我们也根据现有的算法，结合本题实际提出了一种改进版本的扫描线填充算法，从而完整的解决了这个问题。这些是此模型的优点。

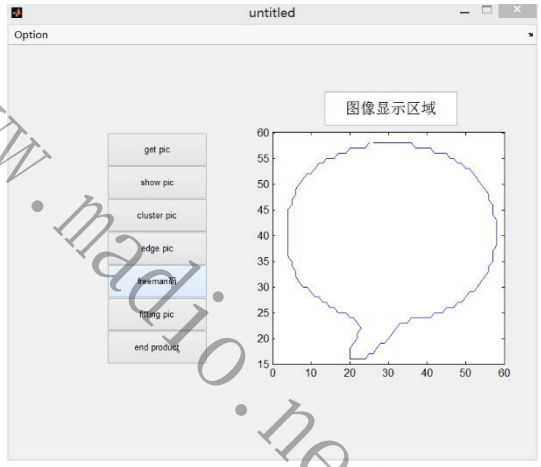
针对此问题的提出的特殊拟合方法较为完美，但是其中相对的普适方法还可以有改进的空间，对于轮廓边缘的提取还可以尝试其他的组合方式，得出更多的比较结果。这些是模型的缺点。

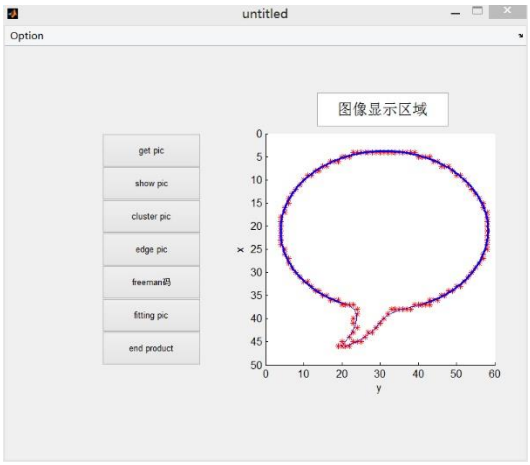
模型的改进方向[15]，从边缘提取来说，我们可以尝试使用一步的方法得到边界数据。从边界数据拟合来说，我们可以尝试多边形近似算法，也可以使用最近研究较多的水平集的方式，从填充区域颜色来说，对于封闭图形，我们可以尝试多图层染色的方式，这样在得到数据点的情况下，填充更加精准，更加高效。由于我们的模型解决的是一个实际应用的问题，虽然在文章的后面提到了一些模型的应用推广，但是还不够，如果能够实现模型的普适性和自动化，将是这种模型的一种重大应用。

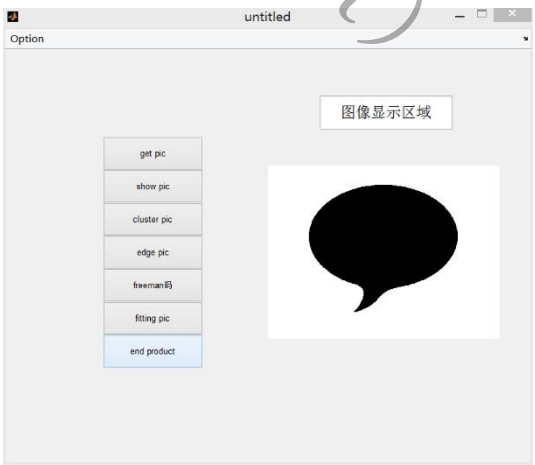
参赛队号 # 1069

应用扩展

这次的题目是研究位图的处理算法，重点是找到栅格化逆过程的办法，在这次题目研究中我们发现，现在市面上很多矢量化的转换软件，所以在这次的研究过程中，有意识朝两个方向发展，一个是程序的集成化，即将程序尽可能的封装；一个是程序的交互性，便于向外人展示。由此我们设计了一个简单的 matlab gui 界面来展示我们整个算法的实现过程，这样做可以使得成果展示更加直接，更好的体现数学模型的思路与使用。







参考文献

[1] 刘玉兰. 光栅图像矢量化技术研究[D]. 首都师范大学 2005
[2] 严素蓉,朱桂林,徐从富. 一种位图矢量化新方法[J]. 计算机工程与应用. 2005(14)
[3] 章孝灿, 潘云鹤. GIS 中基于“栅格技术”的栅格数据矢量化技术[J]. 计算机辅助设计与图形学学报. 2001(10)

[4] 徐国保,王骥,赵桂艳,尹怡欣,谢仕义. 基于数学形态学的自适应边缘检测新算法[J]. 计算机应用. 2009(04)
[5] John Canny. A Computational Approach to Edge Detec-tion. 1EEE Trans. Pattern Analysis and Machine In-telligence. 1986
[6] 周正杰, 王润生. 基于轮廓的形状特征提取与识别方法[J]. 计算机工程与应用. 2006(14)
[7] 井艾斌,柳青,孟祥增. 基于 Matlab 的图形轮廓提取及填充[J]. 电脑知识与技术. 2008(09)
[8] 冯新宇, 方伟林,杨栋. 基于中值滤波与Sobel、Canny算子的图像边缘检测研究[J]. 黑龙江水专学报. 2009(01)
[9] 英英. 基于 MATLAB 的图形图像处理系统的实现[D]. 内蒙古大学 2013

[10] 柳朝阳. 对区域填充扫描线算法的改进[J]. 计算机工程. 1994(S1)
[11] 刘相滨, 胡峰松,张邦基. 一种新的区域种子填充算法[J]. 计算机工程与应用. 2002(08)
[12] 张荣国, 刘焜. 新区入栈的区域填充扫描线算法[J]. 计算机工程. 2006(05)
[13] 王三福, 李莉, 张念喜. 对区域填充算法的一点改进[J]. 天水师范学院学报. 2006(02)
[14] 霍宏涛主编.数字图像处理[M]. 北京理工大学出版社, 2002
[15] 田玉敏,刘国景. 光栅图形矢量化方法分析与评价[J]. 计算机应用研究. 2002(03)

附录：程序

1.图像的获得:

```
function ge=getpic(pic)

    [FileName,PathName]=uigetfile('*');
   pic=imread([PathName,FileName]);
```

2.OSTU 算法实现:

```
function cl=clusterp(pic)

a=pic;
count=imhist(a);
[m,n]=size(a);
N=m*n;
L=256;
count=count/N;
for i=1:L
if count (i)~=0
st=i-1;
break;
end
end
for i=L:-1:1
if count (i)~=0
nd=i-1;
break;
end
end
F=count(st+1:nd+1);
p=st;
q=nd-st;
u=0;
for i=1:q
u=u+f(i)*(p+i-1);
ua(i)=u;
end;
for i=1:q
w(i)=sum(f(1:i));
end;
d=(u*w-ua).^2./(w.*(1-w));
[y,tp]=max(d);
th=tp+p;
for i=1:m
 for j=1:n
 if a(i,j)<th
```

```
 a(i,j)=0;
 else
 a(i,j)=255;
 end
 end
end

imshow(a);
```

3.图像边缘提取：

```
clear all
clc

F=imread('C:\Users\pc\Desktop\gui\图片1.png');
F1=rgb2gray(F);
F2=im2bw(F1);
m1=edge(F2,'canny');
m2=edge(F2,'sobel');
m3=edge(F2,'roberts');
m4=edge(F2,'log');
m5=bwperim(F2,4);


F=rgb2gray(F);
m6=F;
count=imhist(m6);
[m,n]=size(m6);
N=m*n;
L=256;
count=count/N;
for i=1:L
if count(i)~=0
st=i-1;
break;
end
end
for i=L:-1:1
if count(i)~=0
nd=i-1;
break;
end
end
f=count(st+1:nd+1);
p=st;
q=nd-st;
u=0;
for i=1:q
u=u+f(i)*(p+i-1);
ua(i)=u;
end;
for i=1:q
w(i)=sum(f(1:i));
end;
d=(u*w-ua).^2./(w.*(1-w));
[y,tp]=max(d);
```

```
th=tp+p;
for i=1:m
 for j=1:n
 if m6(i,j)<th
 m6(i,j)=0;
 else
 m6(i,j)=255;
 end
 end
end
subplot(2,2,2);
imshow(m6);
subplot(2,2,3);m6=edge(m6,'canny');
imshow(m6);


m11=imcomplement(m1);
m21=imcomplement(m2);
m31=imcomplement(m3);
m41=imcomplement(m4);
m51=imcomplement(m5);
[L,W]=size(m51);
m51(:,1)=1;m51(:,W)=1;m51(1,:)=1;m51(L,:)=1;
m61=imcomplement(m6);


subplot(2,3,1),imshow(m11),title('canny算子')
subplot(2,3,2),imshow(m21),title('sobel算子')
subplot(2,3,3),imshow(m31),title('roberts算子')
subplot(2,3,4),imshow(m41),title('log算子')
subplot(2,3,5),imshow(m51),title('bwperim')
subplot(2,3,6),imshow(m61),title('ostu+canny算子')
```

4.freeman 码程序：

```
function  lian=lianma(pic)

a=pic;
[m,n]=size(a);
img=a;
aa=0;
bb=0;
xx=[];
yy=[];
nn=0;
ed=[1 0;1 -1;0 -1;-1 -1;-1 0;-1 1;0 1;1 1];
for i=2:m-1
    for j=2:n-1
         if img(i,j)==1 %&& imgn(i,j)==0


            % if sum (sum(img(i-1:i+1,j-1:j+1)))~=9
                ii=i;
```

```
                jj=j;

                % imgn(i,j)=2;
                          aa=aa+1;
                           bb=bb+1;
                           xx(aa)=ii;
                           yy(bb)=jj;
            while 1
                nn=rem((nn+4),8);
                 while 1
                  nn=rem((nn+7),8);
                  tempi=ii+ed(nn+1,1);
                  tempj=jj+ed(nn+1,2);
                   if img(tempi,tempj)==1
                       break;
                   end
                end
                 if tempi==i && tempj==j
                     break;
                 else
                     aa=aa+1;
                     bb=bb+1;
                     xx(aa)=tempi;
                     yy(bb)=tempj;
                     ii=tempi;
                     jj=tempj;
                  end
            end
            break;
         end
    end
    if img(i,j)==1
        break;
    end
end
 x=1:length(xx);
 plot(yy(x),n-xx(x));
```

5.分段拟合图像程序：

```
function  full=nihe(F1)

[BoundaryX,BoundaryY]=find(F5==0);
BB=[BoundaryX,BoundaryY];
az = 90;el = 90;
view(az, el);
[nn,mm]=size(BB);
kk=1;gg=1;
for ii=1:nn
    if BB(ii,1)<=37
```

```
        BB1(kk)=BB(ii,1);
        BB2(kk)=BB(ii,2);
        kk=kk+1;
    end
    if BB(ii,1)>=37
        BB3(gg)=BB(ii,1);
        BB4(gg)=BB(ii,2);
        gg=gg+1;
    end
end
% figure(3)
% plot(BB1,BB2,'.')
BB_fix=[BB1',BB2'];
data = BB_fix;
x11=data(:,1);
y11=data(:,2);
X11 = [x11.^2, y11.^2, x11.*y11, x11, y11];
Y11 = ones(size(x11));
L = regress(Y11, X11);
scatter(x11,y11,'r*')
hold on
PFunction=ezplot(@(x,y)
L(1)*x.^2+L(2)*y.^2+L(3)*x.*y+L(4)*x+L(5)*y-1,[-3.5*max(x11)
max(x11) 0.5*min(y11) 1.2*max(y11)]);
set(PFunction, 'Color', 'b');
set(gca, 'XLim',[0 50])
set(PFunction,'LineWidth',2.0);

SecBBfliter1x=[37,37,37,37,37,38,38,38,38,38,38,39:42,43,43,44
,45,45,45,46,46,46];
SecBBfliter1y=[43:-1:21,19];
Coefficient1=polyfit(SecBBfliter1x,SecBBfliter1y,5);
Fittedval1= polyval(Coefficient1,SecBBfliter1x);
Sec1=plot(SecBBfliter1x,SecBBfliter1y,'r*',SecBBfliter1x,Fitte
dval1);
az = 90;el = 90;view(az, el);

SecBBfliter2x=[37,37,37,37,38:46];
SecBBfliter2y=[20:23,24,24,23,23,23,23,22,20,20];
Coefficient2=polyfit(SecBBfliter2x,SecBBfliter2y,4);
Fittedval2= polyval(Coefficient2,SecBBfliter2x);
Sec2=plot(SecBBfliter2x,SecBBfliter2y,'r*',SecBBfliter2x,Fitte
dval2);
az = 90;el = 90;view(az,el);title('');
```

6.样条拟合程序：

```
function nh(xx,yy,fir,nex)    %递归拟合
bew=fix((fir+nex)/2);
```

参赛队号 # 1069

```
%ye=[];
xxn=[];
yyn=[];
TT=4;
 n=0;
 for t=linspace(0,1,nex-fir+1);
     n=n+1;
 xxn(n)=(2*t*t-3*t+1)*xx(fir)+(4*t-4*t*t)*xx(bew)+(2*t*t-t)*xx(nex);
 yyn(n)=(2*t*t-3*t+1)*yy(fir)+(4*t-4*t*t)*yy(bew)+(2*t*t-t)*yy(nex);
% PP=polyfit(xx(fir:nex),yy(fir:nex),3);
% yyn=polyval(PP,xx(fir:nex));
end
ye2s=sum((xx(fir:nex)-xxn).^2+(yy(fir:nex)-yyn).^2);
 if ye2s>=TT
      nh(xx,yy,fir,bew);
      nh(xx,yy,bew,nex);
 else


    t=0:0.001:1;

xxx=(2.*t.*t-3.*t+1)*xx(fir)+(4.*t-4.*t.*t)*xx(bew)+(2.*t.*t-t)*xx(nex)
;

yyy=(2.*t.*t-3.*t+1)*yy(fir)+(4.*t-4.*t.*t)*yy(bew)+(2.*t.*t-t)*yy(nex)
;
   plot(xxx,yyy,'k-')
%    xx(fir)
%    xx(bew)
%    xx(nex)
%    yy(fir)
%    yy(bew)
%    yy(nex)
   hold on ;az = 90;el = 90;view(az, el);
   % plot(xx(fir),yy(fir),'o');
end
End
```

7.寻找尖点程序

```
  text(yy(29),n-xx(29),'P29');
  text(yy(51),n-xx(51),'P51');
   text(yy(61),n-xx(61),'P61');
     text(yy(75),n-xx(75),'P75');
    text(yy(98),n-xx(98),'P98');
 %text(xx(17),yy(17),'尖点');
   figure;
%
subplot(1,2,1);
  x=1:length(xx);

  plot(x,xx(x));
  %%%%%%%%%%%%%%%%%%%%%%找尖点xx%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
 maxj=0;
 T=2;
 for i=11:length(xx)-10
q(i)=abs(xx(i+10)-xx(i)-xx(i)+xx(i-10));
if q(i)>0
 if q(i)>=maxj
    maxj=q(i);
    k=i;
    continue;
 else
     continue;
 end
else
    if maxj>T
        kk=num2str(k);
        xxk=num2str(xx(k));
        xxkk=['(',kk,',',xxk,')','尖点'];
        text(k,xx(k),xxkk);
        maxj=0;
        continue;
    else
        maxj=0;
       continue;
   end
end
  end

subplot(1,2,2);
   x=1:length(xx);

 plot(x,yy(x));
%%%%%%%%%%%%%%找尖点yy%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 maxj=0;
 T=5;
 for i=11:length(xx)-10
q(i)=abs(yy(i+10)-yy(i)-yy(i)+yy(i-10));
if q(i)>0
 if q(i)>=maxj
    maxj=q(i);
    k=i;
    continue;
 else
     continue;
 end
else
    if maxj>T
        kk=num2str(k);
        yyk=num2str(yy(k));
        yykk=['(',kk,',',yyk,')','尖点'];
        text(k,yy(k),yykk);

        maxj=0;
        continue;
    else
        maxj=0;
```

```
        continue;
    end
end
  end
```

8.区域填色程序：

```
%画出最终的拟合曲线——边界点图（比较）
figure(3)
PFunction1=ezplot(@(x)
Coefficient1(1)*x.^5+Coefficient1(2)*x.^4+Coefficient1(3)*x.^3+Coeffici
ent1(4)*x.^2+Coefficient1(5)*x.^1+Coefficient1(6),[SecBBfliter1x(1)
SecBBfliter1x(length(SecBBfliter1x))]);
set(PFunction1, 'Color', 'b');set(PFunction1,'LineWidth',2.0); hold on
PFunction2=ezplot(@(x)
Coefficient2(1)*x.^4+Coefficient2(2)*x.^3+Coefficient2(3)*x.^2+Coeffici
ent2(4)*x.^1+Coefficient2(5),[min(SecBBfliter2x)
max(SecBBfliter2x)]);hold on
set(PFunction2, 'Color', 'b');set(PFunction2,'LineWidth',2.0);
PFunction3=ezplot(@(x,y)
L(1)*x.^2+L(2)*y.^2+L(3)*x.*y+L(4)*x+L(5)*y-1,[-3.5*max(x)        max(x)
0.5*min(y) 1.2*max(y)]);
set(PFunction3,  'Color',  'b');set(PFunction3,'LineWidth',2.0);set(gca,
'XLim',[0 50]);set(gca, 'YLim',[0 70]);az = 90;el = 90;view(az, el);
%画出间断线段
line([46,46],[19.78,20.69],'linewidth',2.0);
line([37,37],[21.55,21.2755],'linewidth',2.0);title('拟合效果')


%画出最终的拟合曲线——边界点图（比较）
figure(4)
PFunction1=ezplot(@(x)
Coefficient1(1)*x.^5+Coefficient1(2)*x.^4+Coefficient1(3)*x.^3+Coeffici
ent1(4)*x.^2+Coefficient1(5)*x.^1+Coefficient1(6),[SecBBfliter1x(1)
SecBBfliter1x(length(SecBBfliter1x))]);
set(PFunction1, 'Color', 'k');set(PFunction1,'LineWidth',2.0); hold on
PFunction2=ezplot(@(x)
Coefficient2(1)*x.^4+Coefficient2(2)*x.^3+Coefficient2(3)*x.^2+Coeffici
ent2(4)*x.^1+Coefficient2(5),[min(SecBBfliter2x)
max(SecBBfliter2x)]);hold on
set(PFunction2, 'Color', 'k');set(PFunction2,'LineWidth',2.0);
PFunction3=ezplot(@(x,y)
L(1)*x.^2+L(2)*y.^2+L(3)*x.*y+L(4)*x+L(5)*y-1,[-3.5*max(x)        max(x)
0.5*min(y) 1.2*max(y)]);
set(PFunction3,  'Color',  'k');set(PFunction3,'LineWidth',2.0);set(gca,
'XLim',[0 50]);set(gca, 'YLim',[0 70]);az = 90;el = 90;view(az, el);
%画出间断线段
line([46,46],[19.78,20.69],'linewidth',2.0,'Color',[0 0 0]);
line([37,37],[21.55,21.2755],'linewidth',2.0,'Color',[0 0 0]);
set(gca,'xtick',[],'ytick',[]);title('');
xlabel('');ylabel('')

%%涂色
```

```
fig=figure(4);
saveas(fig,'C:\Users\asus\Desktop\111.jpg','jpg');
F5=imread('C:\Users\asus\Desktop\111.jpg');
F5=im2bw(F5);

[L,W]=size(F5);
for ii=1:L
    flag=1;
    jj=2;
    while jj<=W-1
        if F5(ii,jj)==0%白色是1，黑色是0;
           jj=jj+1;
           flag=F5(ii,jj); %设置标志
           while flag==1&&jj<=W-2 %找到第一个边界点
                F5(ii,jj)=0; %填充
                jj=jj+1;
                flag=F5(ii,jj);
           end
           break;
           jj=jj+1;
        else
           jj=jj+1;
        end
    end
end


for ii=2:L
    flag=1;
    jj=W-1;
    while jj>=2
        if F5(ii,jj)==0%白色是1，黑色是0;
           jj=jj-1;
           flag=F5(ii,jj); %设置标志
           while flag==1&&jj<=W-2 %找到第一个边界点
                F5(ii,jj)=0; %填充
                jj=jj-1;
                flag=F5(ii,jj);
           end
           break;
           jj=jj-1;
        else
           jj=jj-1;
        end
    end
end
F5=imcomplement(F5);
F6=F5(69:802,157:986);
%F6=F5(18:28,:);
% F7=F6(69:802,:);
figure(5)
imshow(F6)
```

9.Matlab gui程序

参赛队号 # 1069

```
function varargout = untitled(varargin)

% UNTITLED MATLAB code for untitled.fig
%      UNTITLED, by itself, creates a new UNTITLED or raises the existing
%      singleton*.
%
%      H = UNTITLED returns the handle to a new UNTITLED or the handle to
%      the existing singleton*.
%
%      UNTITLED('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in UNTITLED.M with the given input arguments.
%
%      UNTITLED('Property','Value',...) creates a new UNTITLED or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before untitled_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to untitled_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help untitled

% Last Modified by GUIDE v2.5 20-Apr-2014 11:03:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @untitled_OpeningFcn, ...
                   'gui_OutputFcn',  @untitled_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before untitled is made visible.
function untitled_OpeningFcn(hObject, eventdata, handles, varargin)

hmOP=uimenu('label','Option','position',1)
%      hmOPsub1=uimenu(hmOP,'label','Axis on' );
%      hmOPsub2=uimenu(hmOP,'label','Axis off','enable','off' );
%      hmOPsub3=uimenu(hmOP,'label','Grid on',...
```

```
%           'separator','on','visible','off');
%      hmOPsub4=uimenu(hmOP,'label','Grid off',...
%           'visible','off');
%                                        hmOPsub5=uimenu(hmOP,'label','Box
on','separator','on','visible','off');
%      hmOPsub6=uimenu(hmOP,'label','Box off','visible','off');
%        set(hmOPsub1,'callback',[...
%           'axis on,',...
%           'set(hmOPsub1,''enable'',''off''),',...
%           'set(hmOPsub2,''enable'',''on''),',...
%           'set(hmOPsub3,''visible'',''on''),',...
%           'set(hmOPsub4,''visible'',''on''),',...
%           'set(hmOPsub5,''visible'',''on''),',...
%           'set(hmOPsub6,''visible'',''on''),']);
%        set(hmOPsub2,'callback',[...
%           'axis off,',...
%           'set(hmOPsub1,''enable'',''on''),',...
%           'set(hmOPsub2,''enable'',''off''),',...
%           'set(hmOPsub3,''visible'',''off''),',...
%           'set(hmOPsub4,''visible'',''off''),',...
%           'set(hmOPsub5,''visible'',''off''),',...
%           'set(hmOPsub6,''visible'',''off''),']);
%        set(hmOPsub3,'callback',[...
%           'grid on,',...
%           'set(hmOPsub3,''enable'',''off''),',...
%           'set(hmOPsub4,''enable'',''on''),',...
%           'set(hmOPsub3,''checked'',''on''),', ...
%              'set(hmOPsub4,''checked'',''off''),']);
%        set(hmOPsub4,'callback',[...
%           'grid off,',...
%           'set(hmOPsub3,''enable'',''on''),',...
%           'set(hmOPsub4,''enable'',''off''),',...
%           'set(hmOPsub4,''checked'',''on''),', ...
%              'set(hmOPsub3,''checked'',''off''),']);
%        set(hmOPsub5,'callback',[...
%           'box on,',...
%           'set(hmOPsub5,''enable'',''off''),',...
%           'set(hmOPsub6,''enable'',''on''),',...
%           'set(hmOPsub5,''checked'',''on''),', ...
%           'set(hmOPsub6,''checked'',''off''),']);
%        set(hmOPsub6,'callback',[...
%           'box off,',...
%           'set(hmOPsub5,''enable'',''on''),',...
%           'set(hmOPsub6,''enable'',''off''),',...
%           'set(hmOPsub6,''checked'',''on''),', ...
%           'set(hmOPsub5,''checked'',''off''),']);
%   %              hmOPsub5=uimenu(hmOP,'label','Box  on','callback','box
on','separator','on');
% %     hmOPsub6=uimenu(hmOP,'label','Box off','callback','box off');
    hmOPsub7=uimenu(hmOP,'label','Figure color','separator','on');

hmOPsub71=uimenu(hmOPsub7,'label','Red','ForeGroundColor','r',...
           'Callback','set(gcf,''color'',''red'')','accelerator','r');

hmOPsub72=uimenu(hmOPsub7,'label','Blue','ForeGroundColor','b',...
           'Callback','set(gcf,''color'',''blue'')','accelerator','b');

hmOPsub73=uimenu(hmOPsub7,'label','White','ForeGroundColor','w',...
```

```
'Callback','set(gcf,''color'',''white'')','accelerator','h');

hmOPsub74=uimenu(hmOPsub7,'label','Black','ForeGroundColor','k',...

'Callback','set(gcf,''color'',''black'')','accelerator','k');

hmOPsub75=uimenu(hmOPsub7,'label','Yellow','ForeGroundColor','y',...

'Callback','set(gcf,''color'',''yellow'')','accelerator','y');

hmOPsub76=uimenu(hmOPsub7,'label','Green','ForeGroundColor','g',...

'Callback','set(gcf,''color'',''green'')','accelerator','g');



% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to untitled (see VARARGIN)

% Choose default command line output for untitled
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes untitled wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = untitled_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
global picture
[FileName,PathName]=uigetfile('*');
picture=imread([PathName,FileName]);
picture=rgb2gray(picture);


% f=imread('C:\Users\asus\Desktop\gui\图片1.png');
% m=im2bw(f);
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
global picture
imshow(picture);
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
global picture
clusterp(picture)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
global picture
m=im2bw(picture);
getedge(m);
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
global picture
lianma(picture);


% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)

global picture
F1=im2bw(picture)
nihe(F1)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)

global picture
mm=im2bw(picture)
```

```
product(mm)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on key press with focus on edit1 and none of its controls.
function edit1_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  structure with the following fields (see UICONTROL)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
```