血管的三维重建

摘要

本文以血管的三维重建为研究对象，对 100 张平行切片图像进行分析，利用这些宽、高均为 512 象素的切片，计算管道的半径和确定中轴线方程，并在此基础上画出重建后的血管三维图像，主要内容如下：

对于问题一，计算管道的半径，由于血管表面是由球心沿着某一曲线（称为中轴线）的球滚动包络而成，可以得出结论：切片中包含的最大圆的半径即血管半径，所以问题转化为求每一切片上的最大内切圆的半径。为了便于计算，运用 $Matlab$ $imread$ 函数，将 BMP 格式文件转化为0-1矩阵，然后运用 $edge$、$bwmorph$ 函数确定轮廓和骨架的位置，并求解骨架上每一点到边缘的最短距离。这些最短距离中的最大值即为最大内切圆半径也就是血管半径。最后对所有的半径取平均值，得出结果：

$$\overline{R}=\frac{\sum_{k=1}^{100}R_{(k)}}{100}=29.41666$$

对于问题二，根据问题一中求出的100个圆心坐标及半径求解中轴线方程，运用 $Matlab$ 软件对圆心所形成的曲线进行 n 阶多项式拟合。为使中轴线较为光滑，在 $Matlab$ 拟合工具箱多次试验后，取最高阶次 $n=7$。由于 z 轴值是逐层单调递增的，为简化方程的计算，取 t 为参变量，分别对其投影在 YZ、ZX 平面上进行多项式拟合，最后得到中轴线在平面投影上拟合的曲线方程如下：

$$f=\begin{cases} y(t)=-3.23\times10^{-10}t^{7}+1.169\times10^{-7}t^{6}-1.628\times10^{-5}t^{5}+0.00108t^{4}-0.03526t^{3} \\ +0.5706t^{2}-3.105t+5.243 \\ x(t)=3.061\times10^{-10}t^{7}-9.623\times10^{-8}t^{6}+1.36\times10^{-5}t^{5}-0.6406\times10^{-3}t^{4}+0.01912t^{3} \\ -0.298t^{2}+1.89t-1.63.3 \\ z(t)=t \end{cases}$$

最后根据方程画出中轴线图形，YZ、YX、ZX 平面的投影在拟合工具箱中可以直接得到。

对于问题三，根据问题一、二求出的中轴线的参数方程和100张切片的最大内切圆的半径，运用 $Matlab$ 软件画出血管的三维立体图。

关键词： 血管半径　　中轴线　　$Matlab$ 图像处理　　三维重建

一、问题重述

1.1 问题背景与条件

生物体的外部结构具有繁杂多样性，可以通过肉眼观察，但若想了解去内部错综复杂的结构，就需要借助一定的辅助工具，人们常利用的是分解和合成的方法。其中分解就是将生物体做成切片，而切片就是用一组等间距的平行平面将生物体中需要研究的部位切成簿薄的一片，每一片就是生物体某一横断面的图像，当人们需要了解生物体信息时，再采用合成的方法，利用切片信息重建原物体的三维形态。这种方法在医学和其他领域有着广泛的应用，如要将人体的组织、器官、血管等的三维信息，包含内部错综复杂的结构，完整地存储在计算机中，以现在的技术也是有一定难度的，但若改用存储切片信息，使用时重建再现的方法，则是利用现有技术可以解决的。本文为就在此背景下提出下面的问题。

1.2 需要解决的问题

断面可用于了解生物组织、器官等的形态。例如，将样本染色后切成厚约 $1\mu m$ 的切片，在显微镜下观察该横断面的组织形态结构。如果用切片机连续不断地将样本切成数十、成百的平行切片，可依次逐片观察。根据拍照并采样得到的平行切片数字图象，运用计算机可重建组织、器官等准确的三维形态。

假设某些血管可视为一类特殊的管道，该管道的表面是由球心沿着某一曲线（称为中轴线）的球滚动包络而成。例如圆柱就是这样一种管道，其中轴线为直线，由半径固定的球滚动包络形成。

现有某管道的相继 100 张平行切片图象，记录了管道与切片的交。图象文件名依次为 $0.bmp$ 、$1.bmp$ 、…、$99.bmp$，格式均为 BMP，宽、高均为 512 个象素（$pixel$）。为简化起见，假设：管道中轴线与每张切片有且只有一个交点；球半径固定；切片间距以及图象象素的尺寸均为 1。

取坐标系的 z 轴垂直于切片，第 1 张切片为平面 $z=0$，第 100 张切片为平面 $z=99$。$Z=z$ 切片图象中象素的坐标依它们在文件中出现的前后次序为：

$$(-256,-256,z),(-256,-255,z),\cdots(-256,255,z),$$

$$(-255,-256,z),(-255,-255,z),\cdots(-255,255,z),$$

$$\cdots\cdots$$

$$(255,-256,z),(255,-255,z),\cdots(255,255,z),$$

问题一：计算管道的半径。

问题二：确定管道的中轴线方程及画出中轴线在 XY、YZ、ZX 平面的投影图及三维立体图形。

问题三：根据中轴线及半径对血管进行三维重建。

二、模型假设

结合本题的实际，为了确保模型求解的准确性和合理性，我们排除了一些因素的干扰，提出以下几点假设：

1、所有数据均是准确的，根据像素能够近似地描绘出图形；

2、切片的厚度为一个像素；

3、将血管看作一类特殊的管道，不考虑血管的弹性，即血管的半径为一常数；

4、对切片拍照的过程中不存在误差，数据误差仅与切片数字图象的分辨率有关；

5、中轴线上任意两点处的法截面圆不相交。

三、符号说明

为了便于问题的求解，我们给出以下符号说明：

符号	说明
d_{ijk}	第k副图骨架点i到边缘点j的距离
R_k	第k副图的最大内切圆半径
$\overline{R}$	半径的平均值
C_i	第i张图片的重合度
S	原切片图片的上内点及边界点的集合
T	重新切片得到的内点及边界点的集合

四、问题分析

血管可视为一类特殊的管道，该管道的表面是由球心沿着某一曲线（称为中轴线）的球滚动包络而成。本文的主要工作是求解管道的半径及中轴线方程，绘制中轴线在XY、YZ、ZX平面的投影图并重建血管的三维图像。

对于问题一，要求解管道的半径。根据图片分析，除去图片“弯月”的两端之外，这一图形的宽度是保持不变的，这一宽度就是我们所要求的球的直径$2R$。由于管道的表面是由球心沿着某一曲线（称为中轴线）的球滚动包络而成的，所以可以将问题转化为求球体的半径，也就是过球心的被截圆的半径。题目中只给出了100张管道的切面图，这些二维图形是由无穷多个球被截的圆叠加而成的，基于：1）球的任意截面都是圆；2）经过球心的球截面是所有截圆当中半径最大的圆；3）管道中轴线与每张切片有且只有一个交点，即为球心。所以每张切片的最大内切圆的圆心位于血管的中轴线上，该圆的半径等于血管半径，即问题就可以转化为求切片上最大内切圆的半径。首先，运用$Matlab$读取图片，将BMP格式文件转化为0-1矩阵，通过$edge$、$bwmorph$函数确定轮廓和骨架的位置，通过循环不断搜索计算每张图中骨架上每一点到轮廓上每一点的最短距离，然后取最短距离中的最大值，即为最大内切圆的半径，最后对求出的100个最大内切圆半径取算数平均值减小误差，最后的值即为管道半径。

对于问题二，求中轴线，根据问题一中算出的100个最大内切圆半径和圆心坐标，运用$Matlab$软件对所有圆心坐标所形成的曲线进行拟合，根据坐标所画出的散点图规律，采用n阶多项式拟合方式。根据拟合出曲线的光滑度确定最高阶次n，由于一个z只对应于一个x和一个y，故可分别对其投影在YZ、ZX平面上进行多项式拟合，求出方程$y = f_1(z)$和$y = x = f_2(z)$，则中轴线的空间方程即为上两式的联立便得到了血管管道的中轴线参数函数。根据拟合出的方程，画出中轴线在YZ、YX、ZX平面的投影，拟合曲线及立体图形。

对于问题三，基于问题一、二的求解结果，根据中轴线的参数方程及滚动球的半径，我们运用$Matlab$中的绘图工具，结合每张切片图的球心坐标与球的半径，得到血管的还原三维立体图形。

五、模型的建立与求解

经过以上的分析和准备，我们将逐步建立以下数学模型，进一步阐述模型的实际建立过程。

5.1血管半径的计算

算法步骤：

$step1$：导入数据，运用$Matlab$读取函数，转换存储方式；

*step*2：运用*edge*、*bwmorph*函数确定轮廓和骨架的位置；

*step*3：寻找最大内切圆；

*step*4：求解100张图片的最大内切圆；

*step*5：取100个半径的算术平均值。

具体过程：

由于血管是由半径固定的球滚动包络形成，所以过球心的截圆的半径也就是血管的半径，题中给出了血管被切片机平行切割后的100张切片图像，这100张切片图像是由无穷多个球被截的圆叠加而成的，又由于管道中轴线与每张切片有且只有一个交点，即球心。所以在每张切片上总存在着以交点为圆心的圆，此时半径最大，这最大半径同时也是管道的半径。求出最大半径后，运用循环算法，得出这100张切片的最大半径，最后取算数平均值得出管道的半径。

（1）导入数据，转换存储方式

本题中给了100张图片，虽然易于直接观察，但不利于以数学思维建模，为了减少计算量，并保证数据的准确性，首先对*BMP*格式的图像进行预处理，由于*BMP*格式文件在计算机中是以二进制数进行存储的，图像保存在一个二维的有0或1组成的矩阵中。所以，所以将原来512×512单色*BMP*图像格式运用*Matlab*读取函数转化为二维的0-1矩阵，其中1代表黑色像素点，0代表白色像素点。对于一个512×512像素的图像，每一个像素都有自己的一个确定的坐标。由此，可以找出这100张图片的像素矩阵。

（2）确定轮廓线[1]及骨架[2]的位置

为了在减少计算量的基础上，却又反应图像的基本性质，我们确定切片的轮廓线及骨架的位置，其中轮廓是指在亮度不同的区域之间有一个明显的变化，即明度级差突然变化而形成的；而图像的骨架即图像中央的骨骼部分，是描述图像几何及拓扑性质的重要特征之一。具体位置见下图1、2：

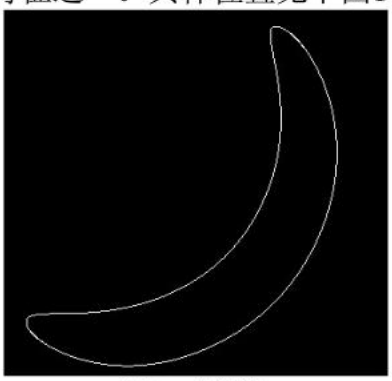


图1　轮廓

图2　骨架

接下来以其中一张图片为例进行详细研究，运用*edge*、*bwmorph*函数确定轮廓和骨架的位置，并用二维矩阵来存储轮廓线及骨架，这样可以减少冗杂的信息量而又突出形状特点，具体方法如下：

在确定骨架时，逐次去掉切片图像轮廓线上的点，同时要保证区域的连通性，余下的部分就是物体的骨架。

运用边缘检测大幅度地减少了数据量，并且剔除了可以认为不相关的信息，保留了图像重要的结构属性。在二维的0-1矩阵中，“0”表示切面上的点，“1”表示切面外的点，对于某一个具体的像素，在其上下左右存在着四个像素，被称为该像素的四邻域，当这一具体像素为“0”时，周围像素中有一个值为“1”，则该点为边界点，用此方法可以对边缘进行检测，确定边缘坐标。

（3）寻找最大内切圆

求出骨架到血管轮廓边界上每一点的距离，则其中的最小距离为即是以该点为圆心的内切圆。找到所有内点的半径后，其中半径最大的内点即为所要找的球心，对应的内切圆即为最大内切圆。若一张切片出现多个圆心时，鉴于所给切片是 *BMP* 格式的图像，是对原血管的切片的连续图形离散化而得到的近似图形，则实际计算时有可能出现在一个切片中同时找到多个最大内切圆，即内切圆不惟一。我们选取与前一张所得圆心距离最近的圆心为当前切片最大内切圆的圆心。

在矩阵内找到轮廓的内切圆记下圆心坐标与半径并储存。骨架看作点的集合，计算骨架上每一点 $A(x,y)$ 到轮廓 $B(p,q)$ 的距离：

$$d_{ijk}=\sqrt{(x-p)^2+(y-q)^2}$$

取 d_{ijk} 最小值，即该点垂直于外边缘，也就是存在着以该点为圆心的圆相切于外边缘。最后取最小距离的最大值，即是最大内切圆的半径：

$$R_k=\max_i \min_j\{d_{ij}\}$$

（4）100张图片的半径

完成骨架上任一点对轮廓线像素点的遍历求距离，并比较记录的所有最小距离值，最小距离的最大值即为该切片图像的最大内切圆的半径，通过循环100次可以得到这100张图片的最大内切圆的半径。运用 *Matlab* 编程（见附录一）得出前40个结果（余下结果见附录二）见下表1：

表 1　最大内切圆半径及圆心坐标

		最大内切圆圆心坐标					最大内切圆圆心坐标		
切片序号	最大内切圆半径 $(R/\mu m)$	横坐标 $(X/\mu m)$	纵坐标 $(Y/\mu m)$	竖坐标 $(Z/\mu m)$	切片序号	最大内切圆半径 $(R/\mu m)$	横坐标 $(X/\mu m)$	纵坐标 $(Y/\mu m)$	竖坐标 $(Z/\mu m)$
0	29.06	-160	1	1	20	29.01	-160	19	21
1	28.28	-160	0	2	21	29.01	-160	20	22
2	29.01	-160	2	3	22	29.01	-160	21	23
3	29.06	-160	2	4	23	29.01	-160	22	24
4	29.06	-160	2	5	24	29.06	-160	21	25
5	29.06	-160	2	6	25	29.06	-160	21	26
6	29.00	-160	1	7	26	29.06	-160	21	27
7	29.01	-160	4	8	27	29.15	-159	30	28
8	29.00	-160	1	9	28	29.27	-159	30	29
9	28.86	-160	1	10	39	29.27	-159	29	30
10	28.86	-160	7	11	30	29.42	-158	35	31
11	28.86	-160	8	12	31	29.61	-157	40	32
12	28.86	-160	9	13	32	29.61	-157	40	33
13	29.01	-160	10	14	33	29.61	-157	40	34
14	29.01	-160	12	15	34	29.61	-156	44	35

15	29.01	-160	13	16	35	29.73	-153	55	36
16	29.01	-160	14	17	36	29.73	-153	55	37
17	29.01	-160	16	18	37	29.73	-153	55	38
18	29.01	-160	17	19	38	29.73	-152	58	39
19	29.01	-160	18	20	39	29.61	-152	58	40

（5）为减小误差，取平均值

从表 1 可以看出，这 100 张图片的最大内切圆半径相近，但也存在着差异，为了使结果更准确，取它们的算术平均值这样可以减少在计算中的误差，计算公式如下：

$$\overline{R}=\frac{\sum_{k=1}^{100}R_{(k)}}{100}=29.41666$$

5.2 中轴线方程

根据表一中求出的 100 个圆心坐标，运用 $Matlab$ 拟合工具箱对圆心所形成的曲线进行拟合，求出中轴线拟合的曲线方程，其中有 4 种拟合方式：线性拟合、抛物线拟合、n 阶多项式拟合和 e^{ax+b} 拟合。前两种拟合方式显然与现实相差太远，而第四种拟合又可以用第 3 种来逼近，所以选择第 3 种方式。对于 n 阶多项式拟合中的最高阶次的取值遵循两个原则：

1）偏差平方和尽量小，这样可以提高数据的拟合度；

2）拟合最高次数不能太高，最好控制在 9 以内，当次数过高时，会增大误差，自变量若有稍微的变动时，应变量的变化会非常大。

根据这两个原则确定 $y(t)$、$x(t)$ 方程的最高阶次 $n=7$，此时拟合出的中轴线较为光滑。以题中给的坐标系为模型，根据空间中轴线是由点组成且 z 轴值是逐层单调递增，为简化方程的计算，取 t 为参变量，分别对其投影在 YZ 、ZX 平面上进行多项式拟合，求出 $y=f_1(z)$ 和 $y=x=f_2(z)$ 最后运用 $Matlab$ 拟合工具箱得到中轴线在平面投影上拟合的曲线方程如下：

$$f=\begin{cases} y(t)=-3.23\times10^{-10}t^7+1.169\times10^{-7}t^6-1.628\times10^{-5}t^5+0.00108t^4-0.03526t^3 \\ +0.5706t^2-3.105t+5.243 \\ x(t)=3.061\times10^{-10}t^7-9.623\times10^{-8}t^6+1.36\times10^{-5}t^5-0.6406\times10^{-3}t^4+0.01912t^3 \\ -0.298t^2+1.89t-1.63.3 \\ z(t)=t \quad ,\ 0\le z\le 99 \end{cases}$$

经检验：$y(t)$、$x(t)$ 方程的拟合度 R^2 分别为：0.9837、0.9935，均大于 0.9，所以拟合度较高。

由此画出中轴线在 YZ、YX、ZX 平面投影拟合曲线及立体图形，见下图 3、4、5、6：

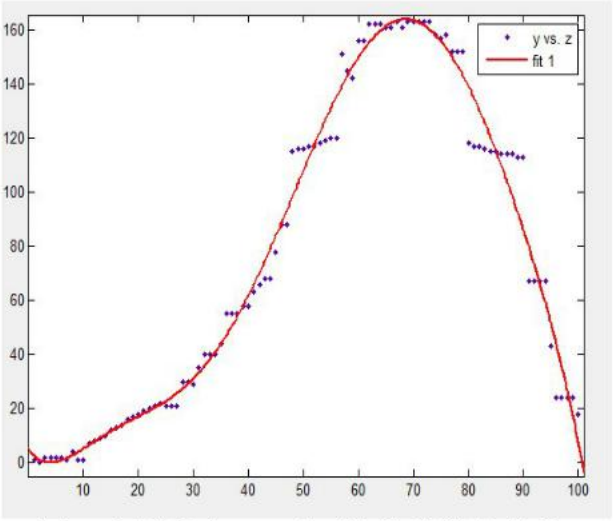


图 3 中轴线在 YZ 平面的投影拟合曲线

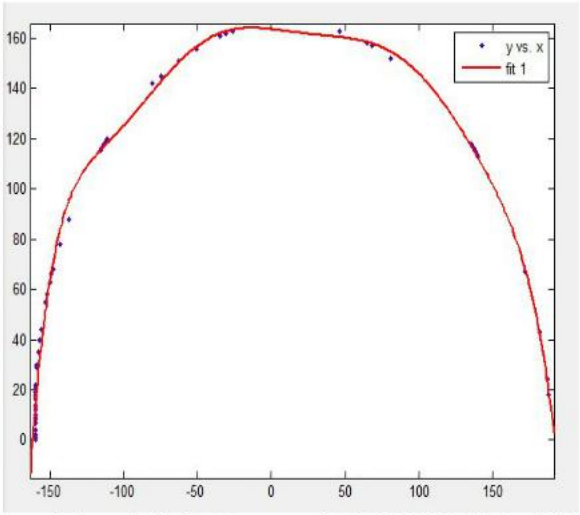


图 4 中轴线在 XY 平面的投影拟合曲线

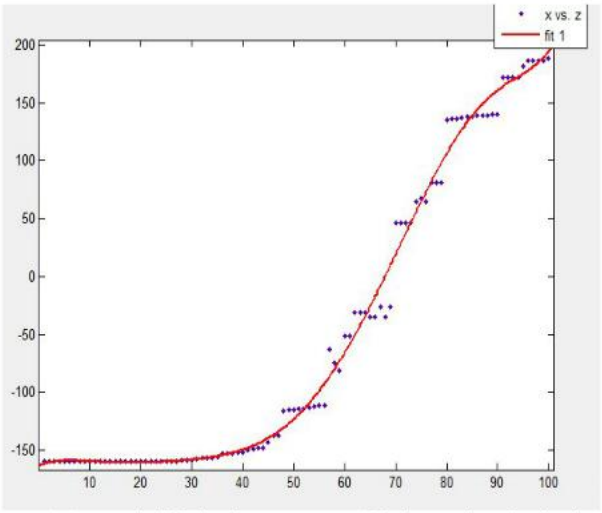


图 5 中轴线在 XZ 平面的投影拟合曲线

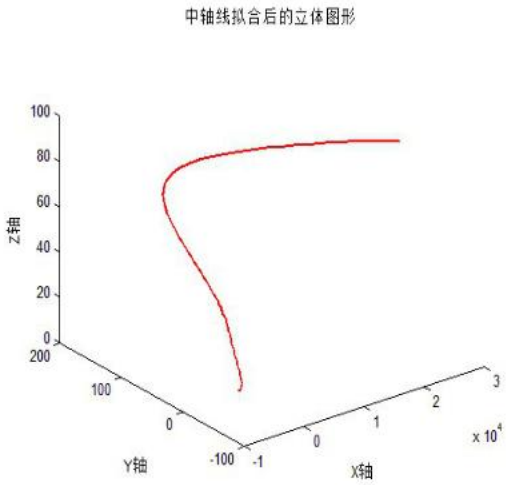


图 6 中轴线拟合的立体图形

分析图像可知：

1)通过三张拟合图发现，得到的曲线都基本经过散点图中各三点的位置，特别在开始的区域，可以说是完全重合，从图案中发现拟合度较高。根据图 4 分析，图形是呈盘旋形状的，随着 x 变大，在 Y 轴变化趋势为：陡—平—陡。结合图 3 和图 5 分析随着 z 的不断变大，图形盘旋上升的趋势为：平—陡—平。

2）拟合的中轴线在整个 z 轴上（$0 \le z \le 99$ ）与血管的实际中心轴基本吻合。

3）尽管所得的差值没有具体的几何意义，但是如果对不同层的差值进行比较，可发现中轴线的拟合在层数 $z<30$ 的地方误差较小，随着 z 值的增大，误差变大，这在直观上是可以理解的；当 z 增大是，轴线与 XY 平面接近平行，$\frac{ds}{dz}$ 变大，切片上容纳的信息有限，产生较大误差。

5.3 重建血管的三维图像

根据中轴线的参数方程及 100 张切片的最大内切圆的半径，运用 $Matlab$ 编程（见附录三）我们得到切片叠加的原图形和血管重建后的三维立体图形见下图 7、图 8：

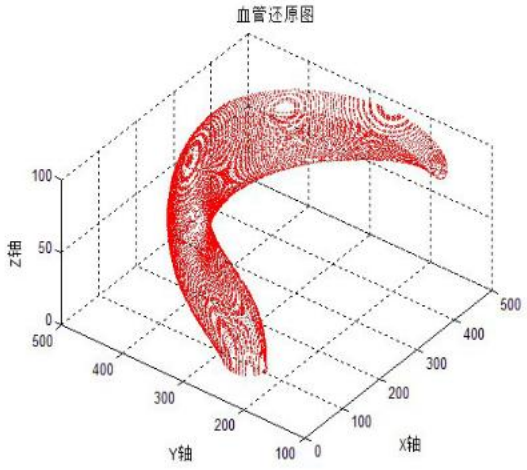


图 7 叠加后血管的还原图

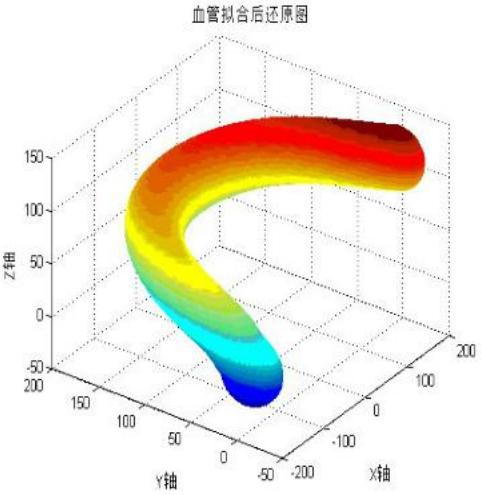


图 8 重建后的血管三维图像

六、模型检验

运用求得的滚动球的半径和中轴线，用球心沿中轴线运动的方法产生一簇球面，其包络面生成一段新的血管。用原来100张的平面截新的血管，生成新的100张横断面图像，抽样比较新、旧100张横截面图像之间的差别。

图 7 给出了血管重建后的三维图像，为了验证这个图像是否与原图相近，而题目中只给出了 100 张切片图，所以将重建后的三维图像，运用 $Matlab$ 切片[3]，随机抽取得到 4 个切片，将新得到的切片与原切片比较，由于无法直接观察，所以我们运用重合度来体现，这可以更有利的证明原切片与新切片的相似程度，计算公式如下（C_i 为第 i 张图片的重合度，S 为原切片图片的上内点及边界点的集合，T 为重新切片得到的内点及边界点的集合）：

$$C_i = \frac{S \cap T}{S}$$

抽取的四张图片分别为：10、20、40、60，图片如下：

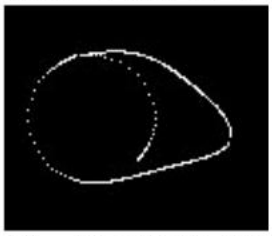


图 a_1 第 10 张拟合图

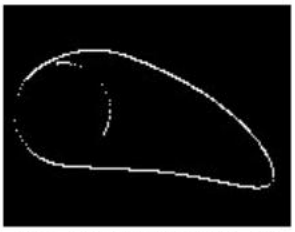


图 b_1 第 20 张拟合图

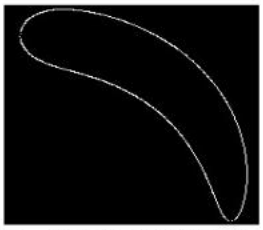


图 c_1 第 40 张拟合图

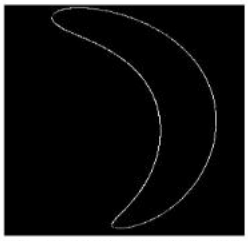


图 d_1 第 60 张拟合图

图 a_2 第 10 张原图

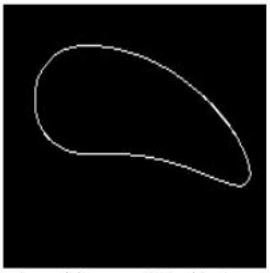


图 b_2 第 20 张原图

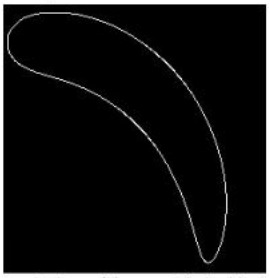


图 c_2 第 40 张原图

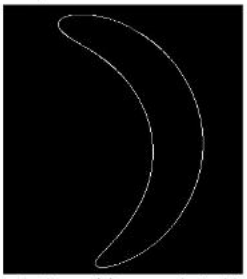
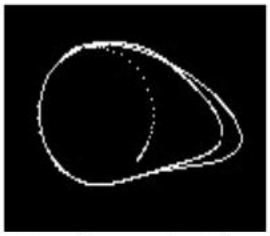
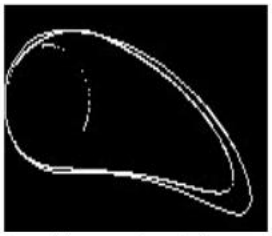
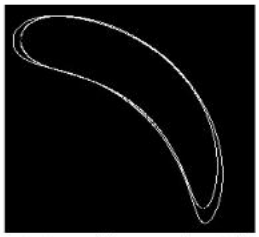
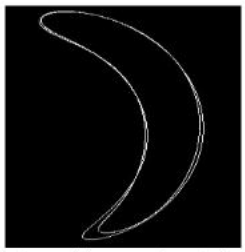


图 d_2 第 60 张原图


图a_3第 10 张叠加


图b_3第 20 张叠加


图c_3第 40 张叠加


图d_3第 60 张叠加

运用$Matlab$编程（见附录四）计算重合度，得出结果如下：

$$C_{10} = 99.2\%$$

$$C_{20} = 98.65\%$$

$$C_{40} = 97.30\%$$

$$C_{60} = 97.61\%$$

这 4 张图的重合度都在95% 之上，即按照我们建立的模型求出来的解与实际的情况基本符合，切面得到的球的所有球的切面构成了整张的切面图形，外轮廓基本重合，这有力的证明了我们的模型的准确性和算法的合理性。所以我们可以得出结论：本模型是可行的，结果较为准确。

七、模型评价与改进

6.1 优点

算法的精确度较高，具有很好的稳定性和可信度，对于半径均匀的血管有很好的适应性，在模型算法的过程中，我们在保证精确度的同时，使算法尽量简单，从而提高编程的运行速度；运用$edge$、$bwmorph$函数确定轮廓和骨架的位置，直观且容易理解；在确定半径时，我们采用平均值法，以减少计算过程中产生的误差；对于中轴线参数方程的确定，7 阶多项式拟合方式，使得曲线较为光滑，视觉效果较好。

6.2 缺点

模型不能够对半径变化的情况适应，这是因为我们的理论根据就是等径管道的几何特性，另外，由于实际中图像边界上的点是连续的，只有位置，没有大小，图像在转换为BMP格式时，像素表示的边界是离散的，产生了系统误差。

6.3 改进

模型的优点体现在算法的精确度上,然而却导致算法的复杂度偏高,因此我们改进的方向主要是在精确度保证的前提下降低复杂度，或者获得精确度和复杂度之间的平衡。在建模的过程中，我们可以发现如果切片的厚度可以进一步缩小，我们得到的拟合曲线的精度更高，反之，对于厚度过大的切片，编程的运行所需的时间会很长，该算法不太适用。对于不规则的三维空间物体，我们的算法不适应，但可在其基础上改进。

八、模型推广

如果考虑中轴线与切片的交点不止一个，比如有两个交点，仍然可以利用我们的离散模型去求解,因为不含轴心的切片中一定不含有最大圆。我们对于离散问题讨论较多，但对于轴心坐标及半径是实数的情况，没有过多的涉及。问题本身是连续的，用离散方法去模拟，不可避免地会出现误差。如果知道象素圆的生成算法，我们就可以运用本模型的思想，将模型推广到实数领域，从而使模型的精度提高。

此模型具有一定的推广意义，尤其是在医学观测领域，医生在不能破坏人体构造前提下,只能通过X 光技术获得断层图像信息，然而仅凭一维的断层图像,医生是很难了解

其内部的复杂结构的，因此，序列图像的三维重现技术却使断层图像得到立体还原，使人体内部构造一目了然。此外,在对精度要求不是很高的很多领域，通过此方法都可以利用计算机实现序列图像的三维重建。

三维实体的重现是当今前沿科技，在医学上有着重要的应用。如胰腺及其周围血管的三维重建和显示及临床意义，此研究实现了胰腺及其周围血管的三维重建和显示,探讨出一种内脏器官和血管的重建方式对断层影像解剖学教学、临床影像学诊断和治疗方案的设计具有明确的辅助作用。基于数字虚拟人体血管的三维重建对教学和临床的应用都有一定实际意义。通过对该模型的改进不仅可以实现血管的三维重现,而且对于复杂管道的测量并重现有着重要的依据。

九、参考文献

[1] 刘惠，图像边缘检测算法分析和实现，湖南师范大学，2001 年 6 月；

[2] 吴德，张红云等，图像骨架提取的应用，2010 年 4 月第四期；

[3] 张德丰等编，*MATLAB* 高级语言编程，机械工业出版社，2010 年 01 月。

十、附录

附录一：

半径求解：

```
clc

jieguo=zeros (100,4);%存储以后生成的结果

for k=0:99
        t=strcat (int2str (k),'.bmp');

        J0=imread (t);%将bmp格式转化为0-1矩阵，黑色为0，白色为1

for i=1:1:512;%像素为512

    for j=1:1:512;

        j0(i,j)=1-J0(i,j); %转化为黑色为1白色为0  为了后面find函数寻找1    find函数是

寻找矩阵中为1的坐标并记录

    end
end

lk=edge(j0,'sobel'); %  提取轮廓          sobel为这个函数中截取轮廓的意思

gj=bwmorph(j0,'skel',inf);%  提取骨架     skel为这个函数中提取骨架的意思

[x0,y0,v0]=find(lk);%找到轮廓灰色区域

[a0,b0,c0]=find(gj); %找到骨架灰色区域

m=length(a0); %轮廓灰色区域个数

n=length(x0);%骨架灰色区域的个数

jl=zeros(m,n); %建立0矩阵为求内切圆半径

cf=zeros(m,2); %建立2列0矩阵为存放中心点坐标

for i=1:m
    for j=1:n
        p1=a0(i);
        q1=b0(i);
        p2=x0(j);

        q2=y0(j); %骨架、轮廓坐标赋值

        jl(i,j)=sqrt((p1-p2)^2+(q1-q2)^2); %  通过循环求各个骨架上的各个点到轮廓的
```

```
距离
    end
    [zx,zxxh]=min(jl(i,:));% 骨架上一点到轮廓的最短距离  即骨架上各个点为圆心的
内切园的半径
    cf(i,1)=zx; cf(i,2)=zxxh;%zx存储小循环内的各个点为圆心的内切园的半径，zxxh存
储最小半径的圆心坐标
end
[zd,zdxh]=max(cf(:,1)); %找到其中最大的半径，并把半径和圆心坐标存储
%%%%%%%%%%%%%%%%%%%%%%
%输出结果
x=a0(zdxh)-256;%与题目所给坐标轴对应
y=b0(zdxh)-256;%与题目所给坐标轴对应
jieguo(k+1,1)=x;%X轴坐标
jieguo(k+1,2)=y;%Y轴坐标
jieguo(k+1,3)=k+1;%Z轴坐标
jieguo(k+1,4)=zd;%半径
end
xlswrite('banjing.xls',jieguo)%导入excle方便使用数据
```

附录二：最大内切圆半径及圆心坐标

		最大内切圆圆心坐标					最大内切圆圆心坐标		
切片序号	最大内切圆半径 $(R/\mu m)$	横坐标 $(X/\mu m)$	纵坐标 $(Y/\mu m)$	竖坐标 $(Z/\mu m)$	切片序号	最大内切圆半径 $(R/\mu m)$	横坐标 $(X/\mu m)$	纵坐标 $(Y/\mu m)$	竖坐标 $(Z/\mu m)$
40	29.54657	-150	63	41	70	29.61419	46	163	71
41	29.54657	-149	66	42	71	29.61419	46	163	72

42	29.52965	-148	68	43	72	29.61419	46	163	73
43	29.52965	-148	68	44	73	29.61419	65	158	74
44	29.52965	-143	78	45	74	29.73214	68	157	75
45	29.41088	-137	88	46	75	29.73214	65	158	76
46	29.41088	-137	88	47	76	29.54657	81	152	77
47	29.69848	-116	115	48	77	29.52965	81	152	78
48	29.69848	-115	116	49	78	29.52965	81	152	79
49	29.69848	-115	116	50	79	29.41088	135	118	80
50	29.69848	-114	117	51	80	29.69848	136	117	81
51	29.69848	-114	117	52	81	29.69848	136	117	82
52	29.69848	-113	118	53	82	29.69848	137	116	83
53	29.69848	-112	119	54	83	29.69848	138	115	84
54	29.68164	-111	120	55	84	29.69848	138	115	85
55	29.20616	-111	120	56	85	29.69848	139	114	86
56	29.41088	-63	151	57	86	29.69848	139	114	87
57	29.52965	-75	145	58	87	29.69848	139	114	88
58	29.52965	-81	142	59	88	29.69848	140	113	89
59	29.54657	-51	156	60	89	29.68164	140	113	90
60	29.54657	-51	156	61	90	29.52965	172	67	91
61	29.61419	-31	162	62	91	29.52965	172	67	92
62	29.61419	-31	162	62	92	29.52965	172	67	93
63	29.61419	-31	162	62	93	29.52965	172	67	94
64	29.61419	-31	162	62	94	29.73214	182	43	95
65	29.61419	-31	162	62	95	29.61419	187	24	96
66	29.61419	-31	162	62	96	29.61419	187	24	97

67	29.61419	-31	162	62	97	29.61419	187	24	98
68	29.61419	-31	162	62	98	29.61419	187	24	99
69	29.61419	-31	162	62	99	29.42788	188	18	100

附录三：

拟合后三维直线图：

```
% 绘制螺旋线

% 绘制原理：随着时间的延续或z坐标的升高，

t=0:99;
x=3.061*10^-10*(t.^7)-9.623*10^-8*(t.^6)+1.36*10^-5*(t.^5)-0.0006406*(t.^4)+0.01912*(t.^3)-0.2908*(t.^2)+1.89*t-163.3;
y=-3.23*10^-10*(t.^7)+1.169*10^-7*(t.^6)-1.628*10^-5*(t.^5)+0.00108*(t.^4)-0.03526*(t.^3)+0.5706*(t.^2)-3.105*t+5.243;
z=t;

% 绘制三维螺旋线

figure
plot3(x, y, t,'red','LineWidth',2);

xlabel('X轴');

ylabel('Y轴');

zlabel('Z轴');

title('中轴线拟合后的立体图形');
```

拟合后三维立体图：

```
format long

b=xlsread ('banjing.xls');%读入中心坐标

x=b(:,1);y=b(:,2);z=b(:,3);%存放拟合后系数

px=polyfit(z,x,7);%通过matlab拟合工具箱发现zx平面7次拟合已经足够，拟合效果足够。

 x1=polyval(px,z);%取拟合后在Z时的点

py=polyfit(z,y,7);%%通过matlab拟合工具箱发现zy平面7次拟合已经足够，拟合效果足
```

够。

```
 y1=polyval(py,z);%取拟合后在Z时的点

cenn=zeros(100,3);%为存储中轴线坐标开辟3列0矩阵

cenn(:,1)=x1;cenn(:,2)=y1;cenn(:,3)=z;%存入坐标
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%以下为画出中轴线为圆心R为半径化圆的血管还原图

t=linspace(0,pi,25);%！！！存储平均分pi为25份储存

p=linspace(0,2*pi,25);%！！！存储平均分2pi为25份储存

[theta,phi]=meshgrid(t,p);%！！！存储角度

for i=1:100
    x=29.4166*sin(theta).*sin(phi)+cenn(i,1);
    y=29.4166*sin(theta).*cos(phi)+cenn(i,2);
    z=29.4166*cos(theta)+cenn(i,3);

    hold on%使图像不被覆盖

    surf(x,y,z)%使图像变为彩色

    %plot3(x,y,z,'red');%使图像变为红色，但是看不清楚，所以改用彩色

    axis([-200 200 -50 200 -50 150]);%坐标轴范围

end

xlabel('X轴');

ylabel('Y轴');

zlabel('Z轴');

title('血管拟合后还原图');

hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

shading interp%使图形颜色变化走势更加光滑
```

附录四：

模型检验：

1、转化矩阵

```
function b0=zhuanhua(b0)      %图像二值矩阵的0-1互换
for i=1:512
    for j=1:512
        if b0(i,j)==1
            b0(i,j)=0;
        else b0(i,j)=1;
        end
    end
end
```

2、重新切片并得到切片后的矩阵

```
function pnjj=dian(px,py,pn) %输出pnjj，为第pn张拟合图片的轮廓二值矩阵
a=zeros(991);
b=zeros(991);
q=zeros(991);
w=zeros(991);
r=zeros(1,2);
s=zeros(1,2);
k=1;
px=[3.06147904917361e-10,-9.62347219220400e-08,1.13641816516904e-05,-0.0006406453
49803698,0.0191190776680884,-0.290773171448047,1.89045866870677,-163.27753640989
8];
py=[-3.23015724809043e-10,1.16915397364673e-07,-1.62765940849020e-05,0.0010802221
4295471,-0.0352637577253801,0.570635949082998,-3.10475858870787,5.24344620268273
];
for c=0:0.1:99
a(k)=7*px(1)*c.^6+6*px(2)*c.^5+5*px(3)*c.^4+4*px(4)*c.^3+3*px(5)*c.^2+2*px(6)*c+px(
7);
b(k)=7*py(1)*c.^6+6*py(2)*c.^5+5*py(3)*c.^4+4*py(4)*c.^3+3*py(5)*c.^2+2*py(6)*c+py(
7);      %中心轴线方程关于z的导数即[a(k),b(k),1]为z在k处的切线的方向向量
q(k)=px(1)*c.^7+px(2)*c.^6+px(3)*c.^5+px(4)*c.^4+px(5)*c.^3+px(6)*c.^2+px(7)*c+px(8);
w(k)=py(1)*c.^7+py(2)*c.^6+py(3)*c.^5+py(4)*c.^4+py(5)*c.^3+py(6)*c.^2+py(7)*c+py(8)
;       %中心轴线方程在z=k处的x,y值
k=k+1;
end
%提取新的截痕
u=[];
v=[];
syms x y
```

```
k=1;
for i=0:0.1:99
    m=a(k)*(x-q(k))+b(k)*(y-w(k))+(pn-i);
    n=(x-q(k))^2+(y-w(k))^2+(pn-i)^2-29.41666^2;
    [g,h]=solve(m,n);
    r=double(g);
    s=double(h);

if (abs(imag(r(1)))<0.01) %去除复数根

    u=[u;[real(r(1))+256 real(r(2))+256]];
    v=[v;[real(s(1))+256 real(s(2))+256]];
end
k=k+1;
end

%根据新的切平面的轮廓坐标得到新轮廓的图像矩阵

plot(v(:,1),u(:,1),'r.',v(:,2),u(:,2),'r.')
axis([0 512 0 512]);
pnj=imread(strcat(int2str(pn),'.bmp'));
lk=edge(pnj,'sobel');
pnjj=zeros(512);
u=round(u);
v=round(v);
for t=1:length(u(:,1))
pnjj(u(t,1),v(t,1))=1;
pnjj(u(t,2),v(t,2))=1;
end

figure(1);%画原图轮廓

imshow(lk)

figure(2);%画新图轮廓

imshow(pnjj)

figure(3);%画原图与新图的轮廓图对比

imshow(pnjj|lk)
pnjj=zhuanhua(pnjj);
```

3、计算新旧切片的重合度

```
function baifenbi=baifenbi(pnjj,pn) %输出拟合图与原切片图的重合度

%填充新图

juzheng=pnjj; %pnjj为通过dian.m得到的轮廓边界二值矩阵
```

```
%先填充左边界的右半部分
for i=1:511
    for j=1:511
        if(pnjj(i,j)==0&pnjj(i,j+1)~=0)
            pnjj(i,j+1)=pnjj(i,j);
        end
    end
end
you=pnjj;
%再填充右边界的左半部分
for i=1:512
    for j=512:-1:2
        if(juzheng(i,j)==0&juzheng(i,j-1)~=0)
            juzheng(i,j-1)=juzheng(i,j);
        end
    end
end
zuo=juzheng;
shijijuzheng=you|zuo; %通过矩阵的或运算得到填充后的新图
imshow(you|zuo)
%原图的黑点的个数
biaozhunjuzheng=imread(strcat(int2str(pn),'.bmp'));
nbiao=0;
for i=1:512
    for j=1:512
        if(biaozhunjuzheng(i,j)==0)
            nbiao=nbiao+1;
        end
    end
end
%新图与原图重合部分黑点的个数
chonghegs=0;
for i=1:512
    for j=1:512
        if(biaozhunjuzheng(i,j)==0&shijijuzheng(i,j)==0)
            chonghegs=chonghegs+1;
        end
    end
end
```

%求百分比

baifenbi=chonghegs/nbiao;