2008 高教社杯全国大学生数学建模竞赛

承　诺　书

我们仔细阅读了中国大学生数学建模竞赛的竞赛规则.

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，我们将受到严肃处理。

我们参赛选择的题号是（从 A/B/C/D 中选择一项填写）：_____

我们的参赛报名号为（如果赛区设置报名号的话）：_____

所属学校（请填写完整的全名）：杭州电子科技大学

参赛队员 (打印并签名) ：1. 宋飞杰

2. 张佳喜

3. 司继春

指导教师或指导教师组负责人 (打印并签名)：数模组

日期：2008 年 9 月 21 日

赛区评阅编号（由赛区组委会评阅前进行编号）：

2008 高教社杯全国大学生数学建模竞赛

编 号 专 用 页

赛区评阅编号（由赛区组委会评阅前进行编号）：

赛区评阅记录（可供赛区评阅时使用）：

评阅人										
评分										
备注										

全国统一编号（由赛区组委会送交全国前编号）：

全国评阅编号（由全国组委会评阅前进行编号）：

数码相机定位算法研究

摘要

本文研究数码相机定位中有关系统标定的相关问题。

首先，本文建立了三个坐标系：像素平面坐标系、像物理平面坐标系和相机坐标系。其中像素平面坐标系和像物理平面坐标系是同一个平面针对不同需要而建立的；相机坐标系是一个世界坐标系，它以相机为参照物。

然后针对第一问确定圆心在像平面上的坐标的问题，本文建立了两个子模型：针孔相机模型和确定靶标相对相机位置的模型，然后提出了运用以上两个子模型求解坐标的方法。

在第一个子模型针孔相机模型中，本文对数码相机进行了适当的简化，即把数码相机看成是一个针孔相机的结构，利用射影几何的有关知识建立了从相机坐标到像物理坐标的转换关系模型。

在第二个子模型确定靶标相对相机位置的模型中，本文利用像平面上四个图形公切线的交点建立了与靶标平面的联系，并结合靶标的尺寸、形状，建立起了确定靶标位置的模型。

在建立了以上两个子模型后，通过第二个子模型可以求出靶标上圆心在相机坐标系中的坐标，再利用第一个子模型的转换关系，就可以得到圆心在像平面上的坐标。

针对模型的求解，本文使用模拟退火算法计算出了像平面上四条公切线交点的坐标，并使用基于最小二乘法的 Matlab 优化工具箱的工具求解出靶标的位置，进一步求出了圆心在像平面上的坐标，五个坐标分别为：A_0(-190.26,-196.77),B_0 (-88.88,-189.15),C_0 (129.74,-172.72),D_0 (72.85,119.30),E_0 (-229.13,119.21)

在模型的检验模型中，本文分别讨论了以上模型的精度和稳定性。在精度检验中，我们将像平面上未被利用的图形的轮廓上的点映射回靶标平面上，并在靶标平面上检验这个轮廓是否与相应的圆形重合。经过检验，轮廓上的点与相应的圆形之间的平均偏差在 1 像素以内，说明以上模型的精度很高。

在随后的稳定性检验中，我们通过计算机模拟的方式，随机改变了像平面上图形的轮廓，并对这些轮廓求解圆心，结果即使在轮廓损失了近 30%的信息量时，圆心的平均偏移距离也只有 0.2682，不到一个像素，说明模型具有很好的稳定性。

最后，本文通过改变世界坐标系，以靶标作为参照物，给出了计算两台相机光学中心、像平面中心坐标的方法，得出了两台相机相对位置的模型。

关键词： 系统标定 射影几何 针孔成像模型 模拟退火算法

一、 问题重述

数码相机定位在交通监管（电子警察）等方面有广泛的应用。所谓数码相机定位是指用数码相机摄制物体的相片确定物体表面某些特征点的位置。最常用的定位方法是双目定位，即用两部相机来定位。对物体上一个特征点，用两部固定于不同位置的相机摄得物体的像，分别获得该点在两部相机像平面上的坐标。只要知道两部相机精确的相对位置，就可用几何的方法得到该特征点在固定一部相机的坐标系中的坐标，即确定了特征点的位置。于是对双目定位，精确地确定两部相机的相对位置就是关键，这一过程称为系统标定。

标定的一种做法是：在一块平板上画若干个点， 同时用这两部相机照相，分别得到这些点在它们像平面上的像点，利用这两组像点的几何关系就可以得到这两部相机的相对位置。然而，无论在物平面或像平面上我们都无法直接得到没有几何尺寸的“点"。实际的做法是在物平面上画若干个圆（称为靶标），它们的圆心就是几何的点了。而它们的像一般会变形，所以必须从靶标上的这些圆的像中把圆心的像精确地找到，标定就可实现。要求给出求解圆心的算法以及方法精度和稳定性的检验，最后讨论两部固定相机的相对位置关系。

二、 符号说明

O_p	像素平面坐标系原点	u	像素平面坐标系横轴
v	像素平面坐标系纵轴	Or	像物理平面坐标系原点
x_r	像物理平面坐标系横轴	y_r	像物理平面坐标系纵轴
O	相机坐标系原点	x	相机坐标系横轴
y	相机坐标系纵轴	M	投影矩阵
R	旋转矩阵	t	平移矢量
K	相机标定矩阵	**AB**（粗体）	向量 AB，粗体表示向量

三、 问题分析

题目第一问要求建立确定靶标上圆的圆心在像平面的坐标的模型。由于存在畸变，所以像平面的图形并不是规则的形状，单纯从像平面的图形出发显然不能找到这个圆心在像平面的坐标。题目中给出了靶标的尺寸等信息，但是靶标像上的点却很难与靶标上的点建立起对应关系。因此我们需要将模型简化，简化成一个小孔成像的模型，这样就可以很容易的将五个圆的外公切线的交点对应起来。对应起这四个点之后，配合靶标的

尺寸、形状等信息，我们就可以确定靶标在三维空间中与相机的位置关系。

确定了靶标与相机的位置关系后，就可以很容易的将靶标上的点投射到靶标像平面上，当然也包括五个圆心。在求取出五个圆心的空间坐标之后，将其投射到像平面上，就得到了第一问需要求的坐标。

在求解时涉及到一个问题，就是像平面上怎样确定切线。因为像平面上的图形是不规则的，所以很难确定这些形状的切线。因此我们考虑另外的方法，使用搜索的办法，利用模拟退火算法求解。

在如何检验模型的问题上，需要分两方面进行检验，一是精度，而是稳定性。

按照以上的方法求圆心在像平面上的坐标，并没有充分利用像平面上所有轮廓点的信息，因此可以利用这些点来检验模型的精度。

对于稳定性问题，可以采用计算机模拟的方法，随机修改图形的轮廓，并用以上的方法再次进行求解，通过比较修改前后的结果来分析模型的稳定性。

最后，考虑另外一台相机的定位相对位置问题。根据前面模型，我们应能够对任意一台相机确定靶标相对它的位置，因此可以以这个靶标作为参照物，建立一个世界坐标系，将这两台相机的位置在这个坐标系里面表示出来，以此确定两台相机的相对位置。

四、 模型假设

1、 假设靶标像的中心恰好在光轴上

2、 假设数码相机中图像平面与光轴垂直

3、 假设相机两个方向上焦距相等

4、 假设透镜的焦距很小，像距约等于焦距

五、 模型准备

(一)靶图像矩阵表示

首先将题目中的图片保存出来，得到的图像可以很方便的放到 Matlab 里面进行处理。但在处理之前还要进行进一步加工：

(1) 将文件读入 Matlab，使用 imread()函数

(2) 将矩阵变为 0-1 矩阵

对于用以上方式得到的矩阵，有两个值：0、15。其中 0 代表像素为白色的点，15 代表像素为黑色的点。为了方便下面处理，对需要把以上像素为 15 的点值全部变为 1。

以上两步的源代码见附录一。

(二)图像轮廓的提取

在提取图像轮廓时，首先要引入计算机图像处理技术中四邻域的概念。

四邻域：某个像素的上、下、左、右四个像素成为该像素的四邻域。如下图所示：

F	B	G
C	A	D
H	E	I

图 1 四邻域示意图

则 A 的四邻域为 B、C、D、E 四个像素。在寻找边界时，对任意一个值为 1 的像素，只要其四邻域有一个值为 0，则认为这个像素为边界上的像素。

此外，经过这样处理后还要剔除孤立的点。方法是若一个像素的周围（包括像素四个角上的像素）值都为 0，则此点为孤立点，予以剔除。

提取图像轮廓的程序见附录二。

提取出的轮廓见下图：

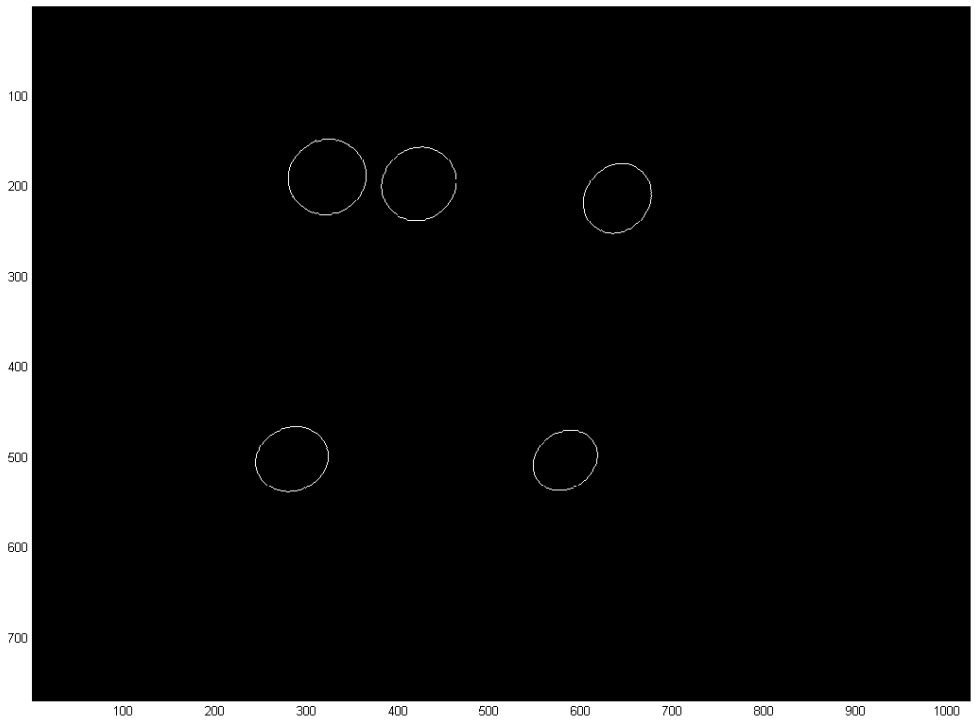

图 2 提取出的轮廓示意图

(三)坐标定义

1. 像素平面坐标系

像素平面坐标即像素的行、列所标识的坐标，如下图所示：

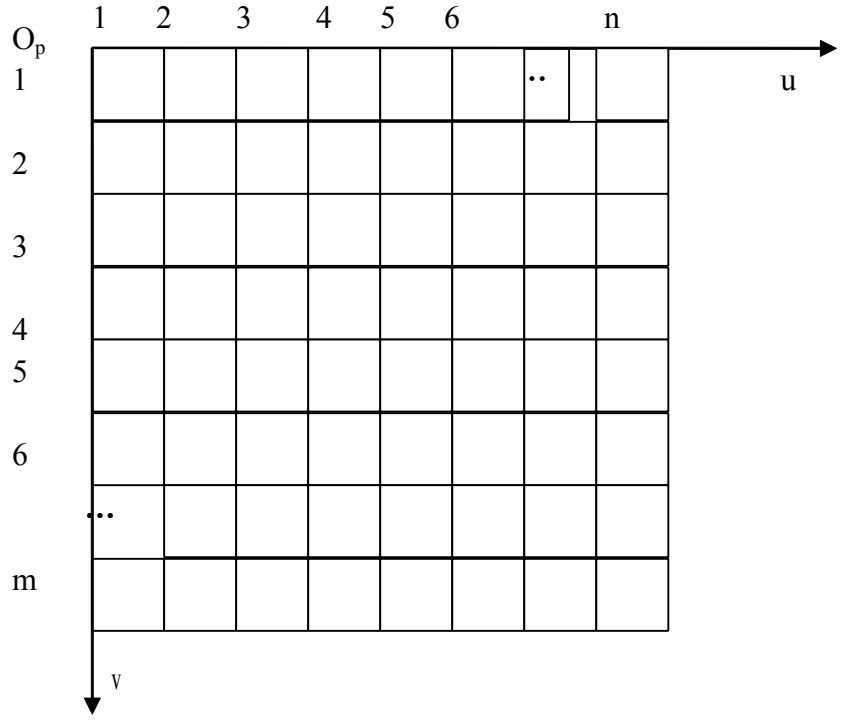

图 3 像素平面坐标示意图

其中每个像素表示为矩阵的行、列数，如(u,v)=(1,1)代表第一行第一列的像素。

2. 像物理平面坐标系

像物理平面坐标系是以图像的中心为坐标原点的坐标系，如下图所示：

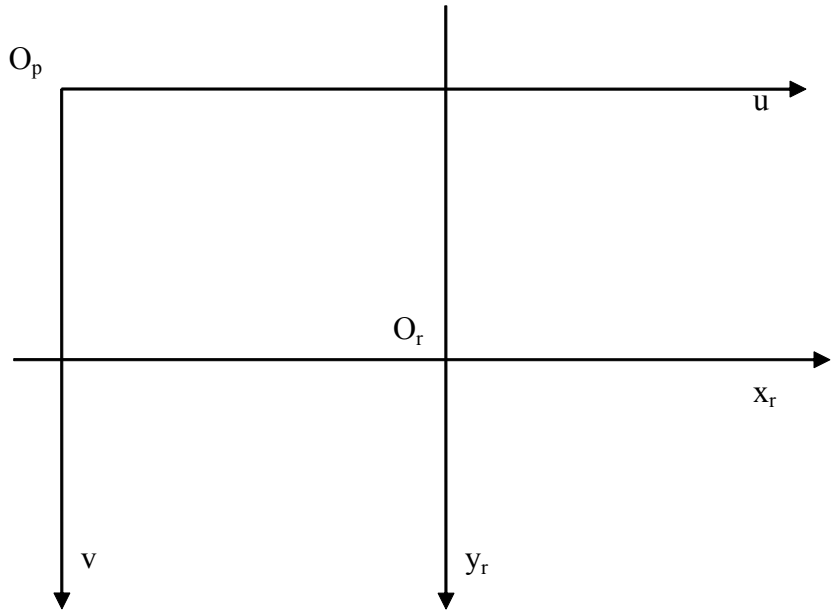

图 4 物理平面坐标示意图

图中 O_p-uv 坐标系为像素平面坐标系，O_r-x_ry_r 为物理平面坐标系。两者的转换关系为[1]：

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{d_x} & s & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} \qquad (1)$$

其中 d_x 和 d_y 为像素点在 x 轴与 y 轴方向上的物理尺寸(在这里使用像素为单位)，u_0 和 v_0 为点 O_r 在像素坐标系中的坐标，s 为图像倾斜度，对于数码相机，可以认为图像倾斜度为一个极小量，一般为 0。[1]

再次，我们选择 d_x 和 d_y 分别为 1，s=0，由于行、列像素数均为偶数，因此选择 u_0=512.5，v_0=384.5，此时，要计算 u、v 时，需要代入像素的中心的坐标来计算。

3. 相机坐标系

相机坐标即整个物理世界的坐标，现根据假设 1、假设 2 做如下坐标：以相机光学中心为原点 O，O 与像的中心 O_r 连成的直线为 z 轴，以过 O_r 平行于像平面 O_r-x_ry_r 且平行于 O_rx_r 的直线为 x 轴，以过 O_r 平行于像平面 O_r-x_ry_r 且平行于 O_ry_r 的直线为 y 轴，建立直角坐标系，如下图所示：

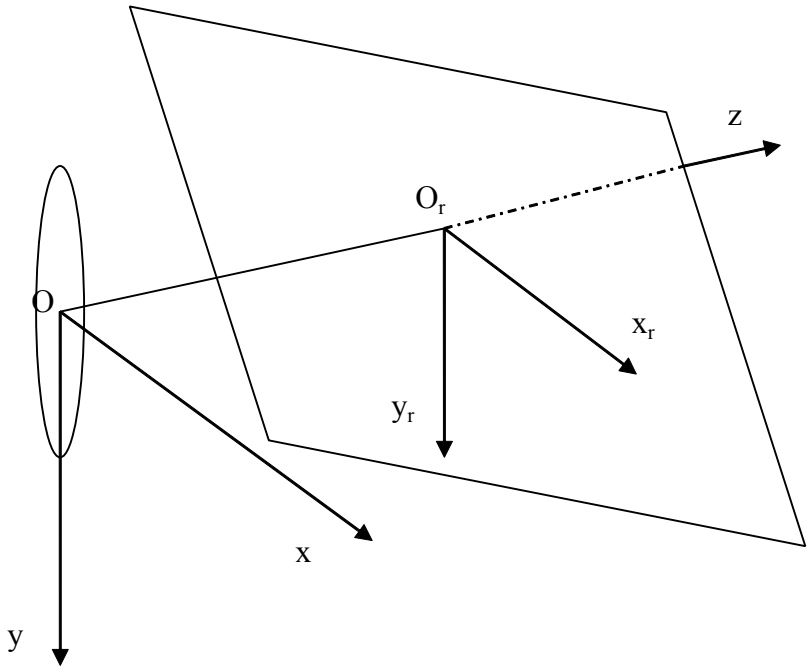

图 5 相机坐标系示意图

(四)分别提取五个圆轮廓的像素坐标

为了便于下面求解，还要将五个圆轮廓的坐标分别提取出来。由于在上一个处理中，已经将图像中可能出现的孤立的点除去，因此现在图像中只存在轮廓上的点，这意味着图像中每一个像素值为 1 的点其四周至少有一个像素的值为 1。因此我们可以使用如下方法提取五个圆轮廓的像素坐标：

i. 令 k=1

ii. 令 i=1，j=1

iii. 判断(i,j)像素值是否为 1，若不为 1，进入第 v 步，否则，进入第 iv 步

iv. 令 i=i+1,j=j+1，跳至第 ii 步。若 i=1024 且 j=768，终止程序。

v. 将(i,j)记录至 B_k，并判断(i-1,j)、(i+1,j)、(i,j+1)、(i,j-1)、(i-1,j-1)、(i-1,j+1)、(i+1,j-1)、(i+1,j+1)是否等于 1，并令(i,j)为以上检验出等于 1 的数值，跳至下一步

vi. 判断(i,j)像素值是否为 1，若不为 1，将(i,j)记录至 B_k，进入下一步，否则，进入第 v 步

vii. 令 k=k+1，对整幅图像剔除孤立点，进入第 ii 步。

通过以上步骤即可得到五个圆的轮廓的坐标（源程序见附录三）。

六、 模型建立

(一)针孔相机模型的建立

相机即一个从三维世界点的集合到二维世界点的集合的一个映射。针孔相机模型是一个没有引入任何非线性畸变的映射，空间点 P 通过一个 3×4 的矩阵映射到影像点 p。

1. 射影几何

射影几何是本文讨论的基础。射影几何是欧氏几何的扩展，解决了欧氏几何中无穷远处元素的问题。

对于三维坐标(X,Y,Z)，其齐次坐标表示为(X_1,X_2,X_3,X_4)，其中 $X=X_1/X_4$，$Y=X_2/X_4$，$Z=X_3/X_4$。对于非零数 λ，$\lambda(X_1,X_2,X_3,X_4)$与(X_1,X_2,X_3,X_4)表示同一个点。若 $X_4=0$，则该点位无穷远点。[2]

2. 针孔相机透视模型

空间点 P 通过一个 3×4 的投影矩阵映射到像平面上的点 p：[3]

$$p = MP$$

投影矩阵 M 可以分为三部分：

$$M = KTG = K\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \qquad (2)$$

其中 R 为旋转矩阵，t 为平移矢量。中间矩阵两个负号代表通过小孔物体呈倒像。K 为相机标定矩阵，它包含相机内部参数数据：

$$K = \begin{bmatrix} f_u & -f_u\cot\theta & u_0 \\ 0 & f_v/\sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (3)$$

其中 5 个参数为：

f_u,f_v：相机焦距，分别以像素为单位沿水平方向和垂直方向测量。根据假设 3、4，这里取 $f_u=f_v=f=1577$。

u_0,v_0：相机主点，即 O_r 点，根据假设 1，这里均为 0。

θ：相机水平与垂直轴向间的夹角，根据假设 2，取 θ 为 π/2。

则标定矩阵可以改写为：

$$K=\begin{bmatrix}-f & 0 & 0\\ 0 & -f & 0\\ 0 & 0 & -1\end{bmatrix}$$

因此，相机坐标(x,y,z)映射到像物理平面坐标(x_r,y_r)就可以写成如下形式：

$$\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}-f & 0 & 0 & 0\\ 0 & -f & 0 & 0\\ 0 & 0 & -1 & 0\end{bmatrix}\begin{bmatrix}R & t\\ 0 & 1\end{bmatrix}\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix} \qquad (4)$$

其中：

$$x_r=\frac{x_1}{x_3}, y_r=\frac{x_2}{x_3}$$

(二)确定靶标平面相对相机位置的模型

通过以上分析，物理世界的点可以通过一次线性变换转换成像物理平面坐标上的点，然而这个过程并不是可逆的。三维空间上的点不能与二维空间上的点建立起一一对应的关系，所以单纯从像平面上的点映射回三维空间上的点是不可能的。

尽管如此，如果我们控制一些参数（比如靶标大小），让这些参数变为已知，或者从像平面上提取一些有用的信息，在这样严格的条件下，仍然可以将像平面上的点近似还原到三维空间中。

针对本问题，我们可以从像平面中提取一些点，这些点可以很容易找到三维空间中与之对应的点，然后根据已知的靶标的长度等数据，就可以近似求出靶标相对于相机的位置。

i. 在靶像上找到四个定点圆的外切线，如下图所示，确定四个交点分别为 A’、B’、C’、D’。

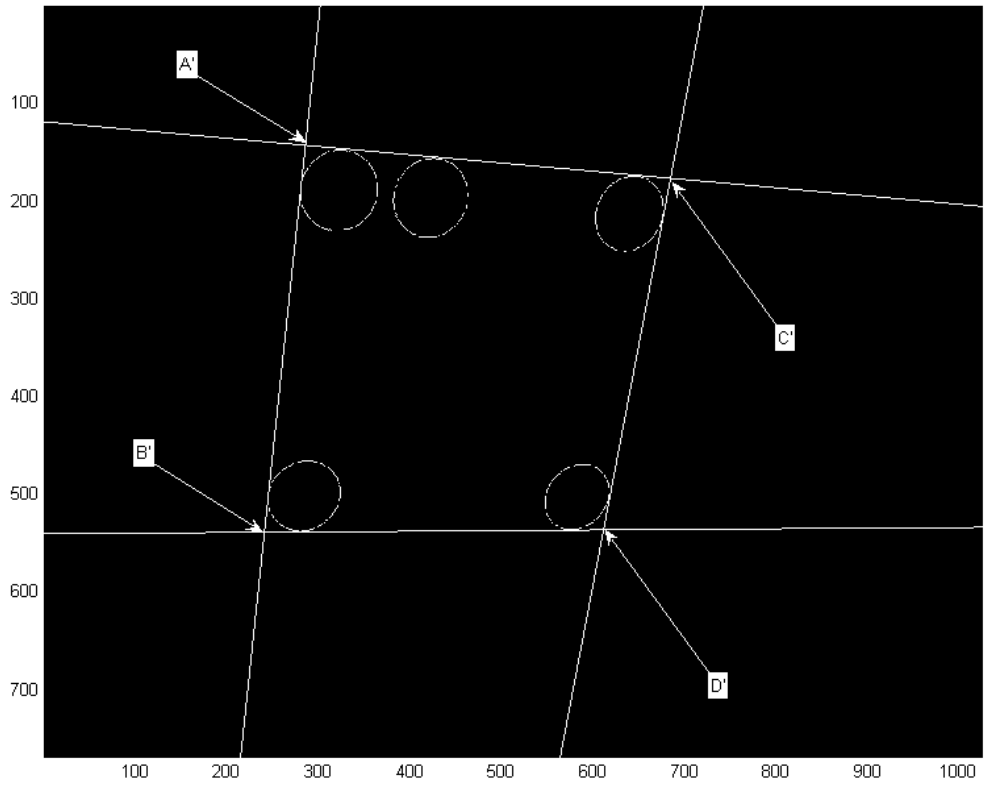

图 6 切线示意图

ii. 将四个点按照如下公式映射到 z=-1 平面上：

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & -1 \end{bmatrix}^{-1} \begin{bmatrix} x_1 \\ x_2 \\ -1 \end{bmatrix} \qquad (5)$$

得到四个新的点 A”、B”、C”、D”。

iii. 得到的四个点 A”、B”、C”、D”既为靶标在 z=-1 平面上的射影，如下图所示：

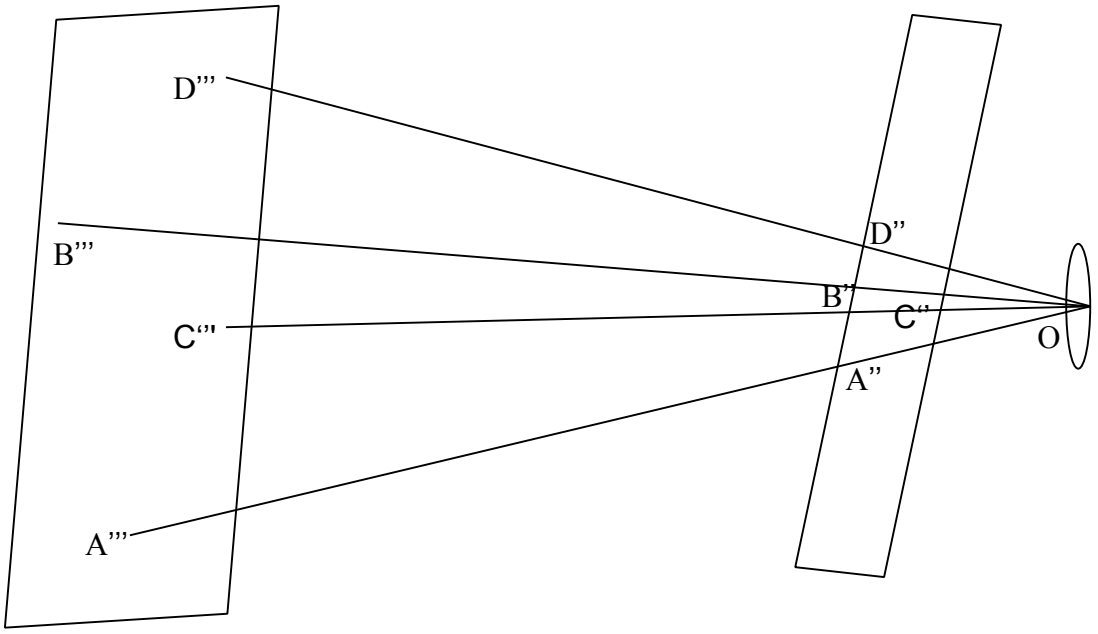

图 7 靶标在 z=-1 上的投影示意图

在靶标上与这四个点相对应的是 A”’、B”’、C”’、D”’四个点，它们是 A、C、D、E 四个圆外公切线的交点，并与 A”、B”、C”、D”一一对应。

由此我们可以设 $\mathbf{OA'''=}\lambda_1\mathbf{OA''}$, $\mathbf{OB'''=}\lambda_2\mathbf{OB''}$, $\mathbf{OC'''=}\lambda_3\mathbf{OC''}$, $\mathbf{OD'''=}\lambda_4\mathbf{OD''}$，进而得到 **A”’B”’**、**A”’C”’**、**B”’D”’**、**C”’D”’**的表达式。这样我们就可以建立以下方程求解参数 λ_1, λ_2, λ_3, λ_4:

$$\begin{cases} \left|\lambda_2\overrightarrow{OB}-\lambda_1\overrightarrow{OA}\right|=l \\ \left|\lambda_3\overrightarrow{OC}-\lambda_1\overrightarrow{OA}\right|=l \\ \left|\lambda_4\overrightarrow{OD}-\lambda_2\overrightarrow{OB}\right|=l \\ \left|\lambda_3\overrightarrow{OC}-\lambda_4\overrightarrow{OD}\right|=l \\ (\lambda_2\overrightarrow{OB}-\lambda_1\overrightarrow{OA})\cdot(\lambda_3\overrightarrow{OC}-\lambda_1\overrightarrow{OA})=0 \\ (\lambda_2\overrightarrow{OB}-\lambda_1\overrightarrow{OA})\cdot(\lambda_4\overrightarrow{OD}-\lambda_2\overrightarrow{OB})=0 \\ (\lambda_3\overrightarrow{OC}-\lambda_1\overrightarrow{OA})\cdot(\lambda_3\overrightarrow{OC}-\lambda_4\overrightarrow{OD})=0 \\ (\lambda_3\overrightarrow{OC}-\lambda_4\overrightarrow{OD})\cdot(\lambda_4\overrightarrow{OD}-\lambda_2\overrightarrow{OB})=0 \end{cases} \qquad (6)$$

其中 l 为靶标上线段 A”’B”’的长度。

需要注意的是虽然只有四个未知数，但是方程仍然使用了 8 个方程，主要原因是这个模型没有考虑畸变的因素，实际的数据带入并不是完全吻合的，所以要在这里求解一个偏差最小的解，所以这 8 个方程都不能舍弃。

利用以上方程组求出 4 个参数之后，就可以确定靶标平面上四个点 A”’、B”’、C”’、

D”’的坐标，靶标平面在相机坐标中的位置也随之确定。

(三)寻找圆心的方法

经过如上处理后，我们就得到了靶标在物理世界中相对于相机的位置。这时只要将靶标平面的圆心映射到相平面上即可找到靶标平面上圆心的位置。映射按照针孔相机透视模型的映射法则进行映射，即：

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (7)$$

由于已经在以上的模型中求出了靶标所在的相对位置，因此其中的 R、t 分别变为单位矩阵和零向量，f 已知，利用这些条件就可以求出像物理平面上圆心的坐标。假设靶标上的圆心坐标为 $O_o(x_o,y_o,z_o)$，则圆心的坐标计算公式为：

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix} \qquad (8)$$

$$\begin{cases} X = \dfrac{x_1}{x_3} \\ Y = \dfrac{x_2}{x_3} \end{cases}$$

(X,Y)既为要求的圆心坐标。

七、 模型的求解

(一)像物理坐标系中切线的求解

给出的像物理坐标系的图像并不是完美的圆形甚至不是椭圆形，给求解公切线带来了一定的难度。并且计算机中的图形表示是离散形式的，这就更加难以处理。为此，我们引入了模拟退火算法（SA）进行求解。

模拟退火算法是一种搜索算法，相比于遗传算法、蚁群算法等算法，这种算法比较容易陷入局部最优。但针对本问题，不存在局部最优问题，因此模拟退火算法仍是一个高效而又准确的算法。

1. 求解思路

求解切线方程并不是最优问题，但是我们可以通过一定方法将其转化成一个最优问题。

首先是约束条件。我们所研究的图形的切线跟图形理论上应该只有一个交点，但是在计算机中，这个形状的边界点是离散的而非连续的，所以很难保证切线与图形只有一个交点，即使在离散的坐标中的确只有一个交点。因此我们这里并不希望切线与图形之间有交点，而是图形与切线的最近的点是相邻的。这样最大的误差就控制在一个像素范围以内。比如在下图中：

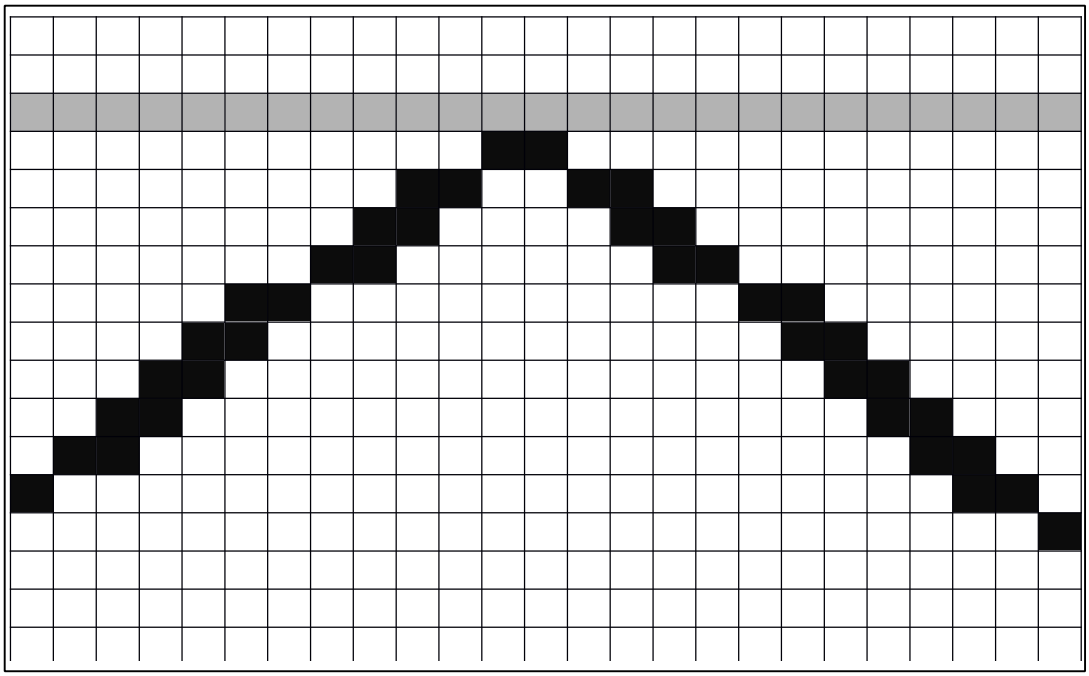

图 8 切线示意图

灰色代表切线，黑色代表图形轮廓。如果切线与形状相交，只会造成更大的误差。

然后是目标函数。由于约束条件选择了让切线不与图形相交，因此目标函数要让切线尽量向图形靠近。我们在这里选取四条切线围成的面积最小为目标函数。在计算四边形面积时，将四边形分为两个三角形，用以下公式计算三角形面积：

$$S=\frac{1}{2}\begin{vmatrix} a & b & 1 \\ c & d & 1 \\ e & f & 1 \end{vmatrix}$$

其中(a, b)、(c, d)、(e, f)为三角形三个顶点的坐标。

此外，用于求解的参数也不直接用四条切线的方程系数，而是用四个交点作为参数进行求解。四个交点坐标可以表示四条切线，并且减少了参数数目，省去了计算交点坐标的步骤。

2. 初始解的给出

在这里给出初始解只是为了加快算法收敛的速度，应用中也可以随机取四个可行的点，此做法不失一般性。

为了给出初始解，我们现在原图中画四条靠近图形但是不与图形相交的直线，然后找到四个交点的坐标，如下图所示：

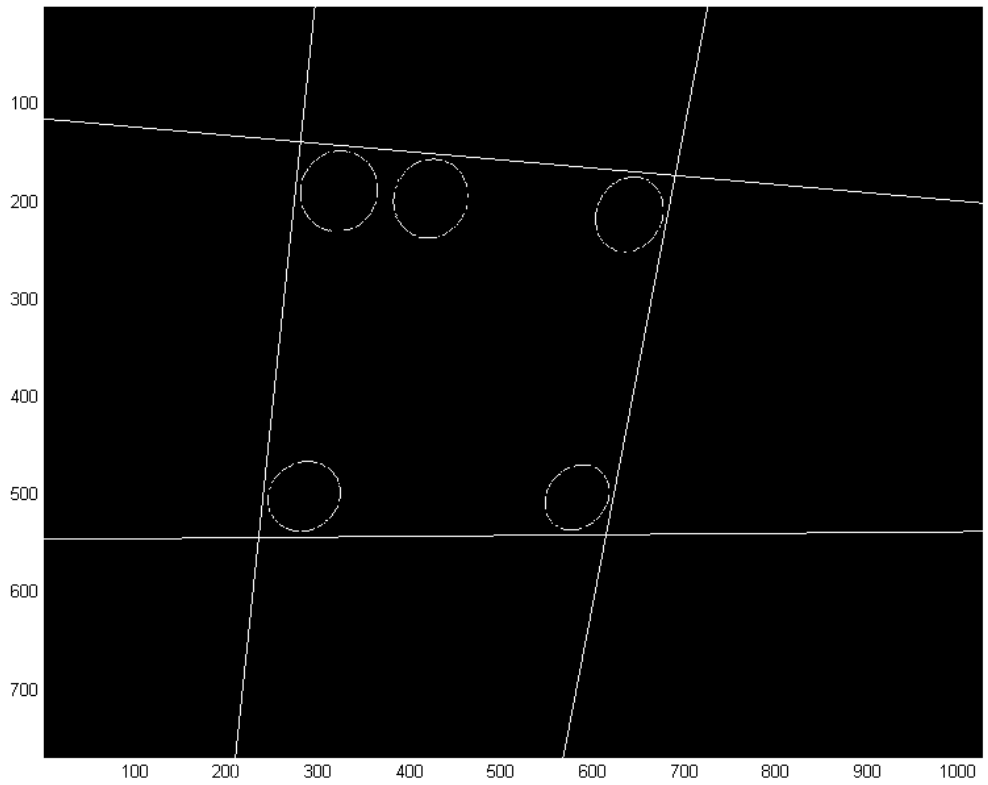

图 9 初始解寻找

由此得到的四个初始解（像素坐标）为：(256,139),(235,546),(690,172),(617,544)。

3. 模拟退火算法过程

i. 设定初始温度 T_0，最终温度 T_1，扰动最大最小范围 L_{max},L_{min}，初始化迭代次数 cs

ii. 令 $T=T_0$ 计算初始解 r_0 的目标函数值 E_0

iii. 产生一个[0,L]内的随机扰动加至 r_0 上，判断加入扰动后是否在可行域上，若否，则继续产生可行域，直到加入扰动后在可行域上，加入扰动后的解为 r_1

iv. 计算 r_1 的目标函数值 E_1

v. 计算 $\Delta E=E_1-E_0$

vi. 若 $\Delta E<0$，则令 $r_0=r_1,E_0=E_1$ 跳至第 viii 步，否则，跳至第 iii 步

vii. 产生一个随机数 u，若 $u<\exp(\Delta E/T)$，则令 $r_0=r_1,E_0=E_1$

viii. 判断迭代次数 k 是否为一个整数常数 K 的整数倍，若是，则令 T=aT，a 为一个常数

ix. 若 $T<T_1$，则进入 x，否则令 k=k+1，跳至第 iii 步

x. 结束

为了防止程序在最后收敛时在可行域边缘产生随机扰动时经常超出可行域，我们在一开始给定了一个最大最小范围 L_{max},L_{min}，在每次迭代时，L 都做如下计算：

$$L=\frac{L_{\max}-L_{\min}}{T_1-T_0}T+L_{\min}-T_{\min}\frac{L_{\max}-L_{\min}}{T_1-T_0}$$

我们选取 T0=100,T1=0.01,L_{max}=10,L_{min}=1,k=10,a=0.9 进行计算（源程序见附录四），得到的结果为：A”(285,143),B”(240,539),C”(685,177),D”(613,537)，如下图所示：

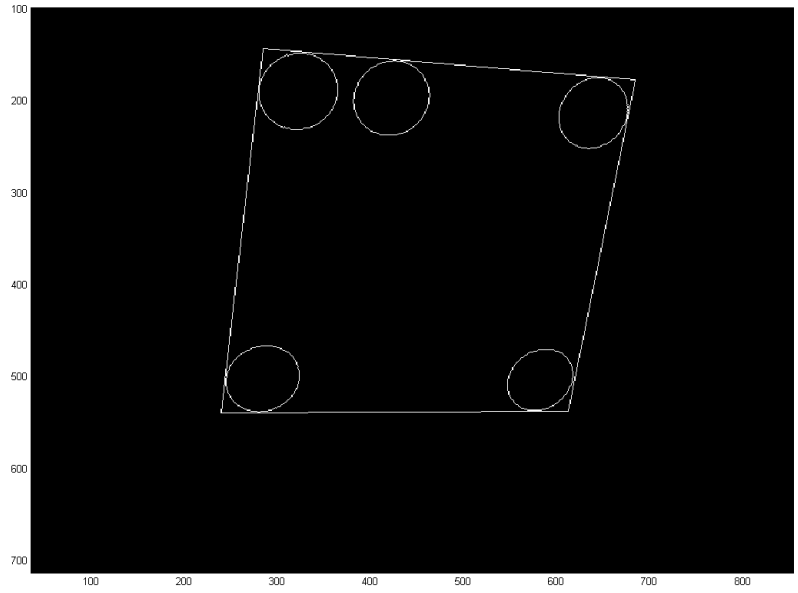

图 10 求解切线结果图（放大后）

(二)圆心坐标的计算

1. 靶标平面位置的确定

前面的模型建立中已经给出了确定靶标平面的方程组的模型。由于像存在畸变，因此方程组很有可能无解。但是可以根据这个方程组求一个近似解，这个近似解也就是我们要取的近似的靶标平面。

具体求法我们使用 Matlab 里面的 fsolve()函数。fsolve 函数是 Matlab 软件优化工具箱中用于求解非线性方程组的函数，其算法基于最小二乘法，可用于求解非线性方程组的零点。如果一个系统能表示成一个非线性方程组，便可用 fsolve 函数求零解。[4]

在使用 fsolve 函数时，初始解的选取非常重要。若初始解不合理，则 fsolve 函数很容易陷入局部最优或者提前结束迭代。我们可以根据像的大小和像距来大体估计靶标平面距离相机的距离：

在像中，四边形的长大约为 400 像素，即大约有 105mm。而靶标的真实长度有 100mm，因此可以得出结论，靶标距离相机坐标 xoy 平面与像距离相机坐标 xoy 平面的距离大体相等。而 O 点到 O_r 点有 1577 像素，因此不妨令初始值为[1500,1500,1500,1500]。

如此，使用 fsolve 函数求解（见附录五），求得：

$$[\lambda_1, \lambda_2, \lambda_3, \lambda_4]=[1657.63,1815.08,1829.54,1984.68]$$

同时计算所得的四条边平均长度误差为 3.19 像素，相对误差 0.844%，误差比较小，可以接受。

因此靶标上四条外公切线交点 A'''、B'''、C'''、D'''的坐标分别为：(239.13,253.85,-1657.63),(313.64,-177.82,-1815.08),(-200.12,240.73,-1829.54),

(-126.48,-191.92,-1984.68)。

用以上四个点中的任意三个就可以表示靶标所在的平面。

2. 圆心的确定

对于下图所示的靶标平面：

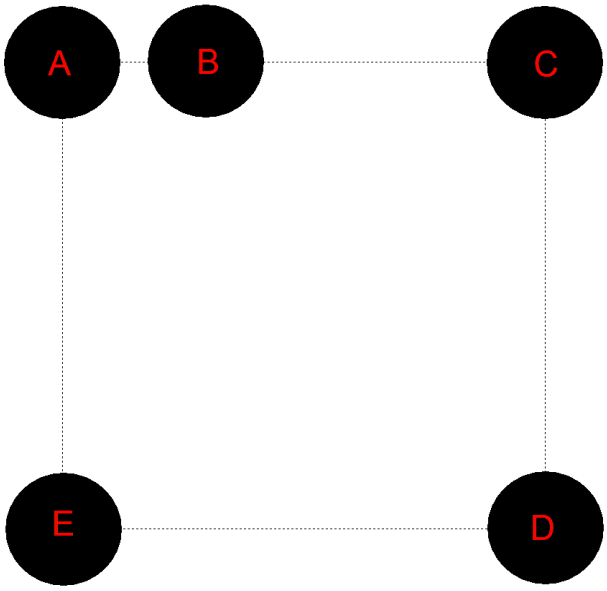

图 11 靶标平面

如果以 **A'''B'''**和 **A'''C'''**作为平面的基向量，则五个圆的圆心分别可以表示为：

$$\begin{aligned}
\overrightarrow{A''A} &= \frac{12}{124}\overrightarrow{A''C''} + \frac{12}{124}\overrightarrow{A''B''} \\
\overrightarrow{A''B} &= \frac{42}{124}\overrightarrow{A''C''} + \frac{12}{124}\overrightarrow{A''B''} \\
\overrightarrow{A''C} &= \frac{112}{124}\overrightarrow{A''C''} + \frac{12}{124}\overrightarrow{A''B''} \\
\overrightarrow{A''D} &= \frac{112}{124}\overrightarrow{A''C''} + \frac{112}{124}\overrightarrow{A''B''} \\
\overrightarrow{A''E} &= \frac{12}{124}\overrightarrow{A''C''} + \frac{112}{124}\overrightarrow{A''B''}
\end{aligned} \qquad (9)$$

同样，在三维空间中，上述表达式依然成立。我们已经知道了 A'''，B'''，C'''三点在相机坐标系中的坐标，因此很容易就可以将 A、B、C、D、E 的坐标表示出来。然后通过真空相机透视模型的转换矩阵就可以很容易的找到像物理平面上对应的圆心的坐标。

经过计算机程序计算（见附录五），得到的圆心坐标为：

（像物理坐标系）A_0(-190.26,-196.77),B_0 (-88.88,-189.15),C_0 (129.74,-172.72)

D_0 (72.85,119.30),E_0 (-229.13,119.21)

（像素坐标系）A_0 (322,188),B_0 (424,195),C_0 (642,212),D_0(585,504),E_0 (283,504)

圆心位置如下图所示：

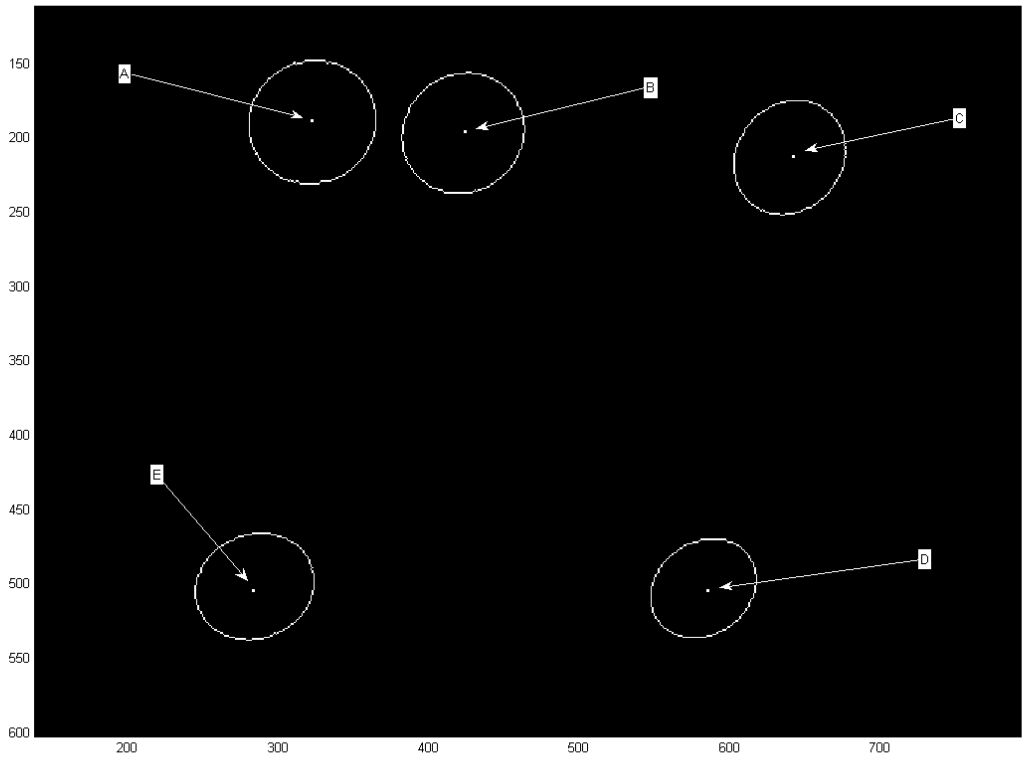

图 12 圆心位置

八、 模型的精度分析与检验

(一)误差产生的原因分析

1. 畸变产生的误差

以上本文讨论的是线性变换的模型，并没有考虑图形产生的畸变。实际的相机拍的照都是有畸变的，这些畸变的产生一部分是由于透镜成像的原因，另一部分是由于空气等的折射因素，此外还有极其复杂的其他原因。以上模型建立在没有畸变的基础上，难免会对精度产生影响。

2. 连续图形离散化产生的误差

被拍摄的物体一般是连续的，然而物体拍摄到数码相机后，原来连续的点被离散化，会丢失很多图像边缘的信息。在以上模型中，求解切线就严重依赖于图像的边缘。而这些图像边缘丢失的信息会影响切线的形状，进而影响求得的靶标的相随位置以及最后求得的圆心。

3. 边缘图像颜色不确定带来的误差

这种误差不同于以上离散化的误差。以上离散化产生的误差是指一个连续的线离散化时需要丢弃一些点，以至于形状改变。而边缘化图像颜色不确定产生的主要原因是拍摄时对焦不精确等原因产生的。实际拍摄中，对焦不可能完全精确，因此在图像中颜色变化之间的点的颜色经常会显示出一些过渡颜色，比如在黑色与白色之间，并不是黑白分明，而是会有一些像素是灰色。

在以上我们的处理中，有一项是在 Photoshop 中用“阈值”工具将图像处理为黑白颜色。然而这往往是不精确的。比如我们选择阈值为 205，当选择 206 时，又有一些像素将被处理成白色，依次类推。阈值无法准确确定，并且也不存在精确的阈值，因此这种操作会给图像边缘带来一定误差。

4. 求解误差

以上所讲所有导致误差的原因都是客观条件上产生的，而求解误差则存在于模型本身当中。比如我们以上的模型是建立在“光线垂直摄入相机的光学中心，出射光刚好射在图像的中心上”等一系列假设条件之上的，这些假设不一定符合事实。在大多数情况下，这些假设总是近似符合事实的（否则模型就会出现不符合实际的错误，而不是误差），这里的“近似”符合事实也会产生误差。

(二)精度检验的基本思路

在以上模型的求解过程中，我们并没有利用图形中给出的全部信息就得到了结果，图像的边缘并没有被完全利用，这些没有被完全利用的信息就是我们建立精度检验模型的基础。

我们以上求解圆心坐标时，是直接以三维空间内靶标上的圆心投射到像上，并以此作为圆心。这里面暗含了一个假设，即像上的点都是由我们求得的靶标位置上的圆的点形成的。但是求得的靶标的位置是不精确的，因此三维空间内圆上的点与像上圆上的点可能会有偏差。在理想情况下，若不存在这种偏差，则我们求得的圆心就是要求的精确的圆心；这种偏差越大，圆心的位置就越不可靠。

因此精度检验可以转换为像上的图形与靶标上的圆的映射关系的检验。

具体方法是，首先将像上的轮廓的点投射到以上模型求得的靶标上，然后计算靶标上相应的圆心距离这些点的平均距离和标准差，平均距离越贴近于靶标上圆的半径，则模型越准确；标准差越小，说明模型越准确。

(三)检验算法及求解

1. 从像物理平面坐标到相机坐标

下面我们考虑一个在像上的点怎样映射到求出的靶标平面上：

如下图所示：

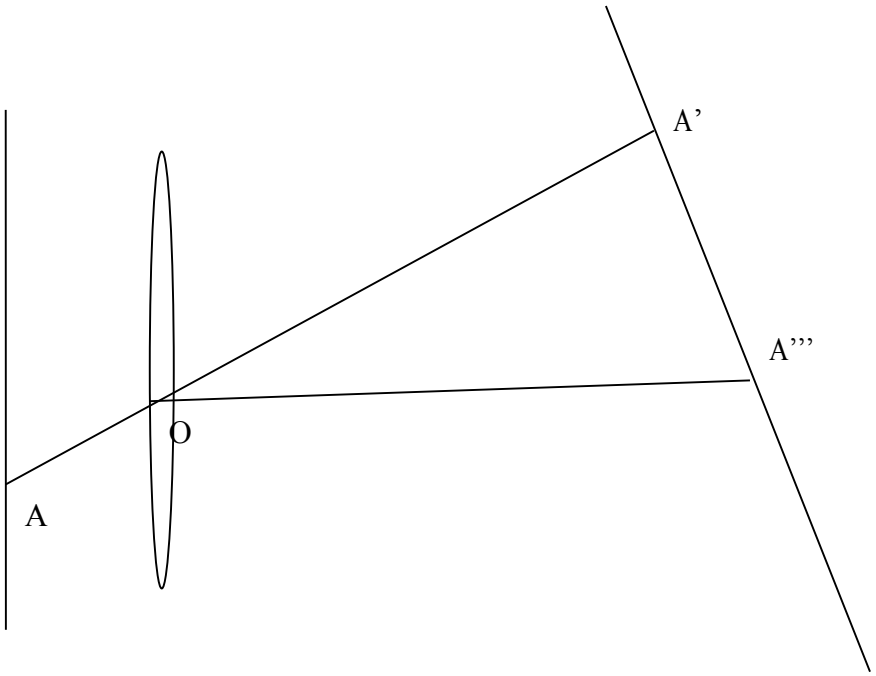

图 13 映射示意图

现假设像平面上任意一点 A 坐标为(a,b,c)，则 **AO**=(-a,-b,-c)，若假设 **OA’**= λ **AO**，那么，**OA’**=- λ (a, b, c)。

现在我们取靶标平面上的点 A’’’(m,n,p)，同时 **A’’’B’’’**=(u_x,u_y,u_z),**A’’’C’’’**=(v_x,v_y,v_z)。则 **OA’** 还可以表示为 **OA’=OA’’’+** λ_1**A’’’B’’’+** λ_2**A’’’C’’’**=(m,m,p)+ λ_1(u_x,u_y,u_z)+ λ_2 (v_x,v_y,v_z)，用矩阵可以表示为：

$$[\lambda \quad \lambda_1 \quad \lambda_2]\begin{bmatrix} a & b & c \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{bmatrix} = -[m \quad n \quad p]$$

要求解 λ，只要计算如下式：

$$[\lambda \quad \lambda_1 \quad \lambda_2] = -[m \quad n \quad p]\begin{bmatrix} a & b & c \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{bmatrix}^{-1} \qquad (10)$$

即可。

这样，就可以得到 A‘的坐标为(－λ a, －λ b, －λ c)。

2. 计算映射点与圆心距离的均值与方差

假设映射到靶标平面上的点位 A’，预期对应的圆心为 O’，则下一步计算两个点之间的距离|O’A’|。对于原边缘的整个序列上的点，计算所有的距离，并计算出所有这些点距离相应圆心的距离的均值 μ 与标准差 σ 。

经过计算（源程序见附录六），得到如下的结果（以像素为单位）：

表一 精度检验结果

圆心	μ	偏差	偏差/半径	σ	标准差系数（σ／μ）
A	45.98	0.62	0.0137	1.90	0.0413
B	46.03	0.67	0.0149	1.63	0.0354
C	46.35	0.99	0.0219	2.26	0.0487
D	46.28	0.92	0.0203	2.49	0.0538
E	45.85	0.49	0.0108	1.90	0.0417

从以上表可以看出，以|O'A'|的平均偏差都不大于一个像素，标准差也控制在 2 像素左右标准差系数非常小，说明像平面上的点映射到靶标平面上，得到的点比较好的拟合了原靶标平面上的五个圆，因此有理由相信，经过以上算法计算得到的像平面上的圆心精度很高。

九、 模型的稳定性模拟与检验

由于种种原因，像的轮廓上的点很多情况下是难以确定的，而这些难以确定的点常常会影响模型求解的结果。一个好的模型不仅要精确，而且在轮廓出现变化时，模型的结果不能变化太大，即要求模型的稳定性要好。

为了检验本模型的稳定性，我们采用计算机模拟的方法来破坏像的轮廓，再采用以上建立的模型求解，比较两个结果就可以判断模型稳定性的好坏。

为了简便起见，我们只考虑轮廓缩小的情况，不考虑轮廓放大的情况。对于轮廓缩小的情况，我们可以随机减少轮廓上的点来模拟图像轮廓的变化。

在轮廓上减少点，利用模拟退火算法求解外公切线交点坐标时，其最优解目标函数值只会比减少钱更小，因此不妨令模拟退火算法的初始点为以上我们模型确定的 4 个点 。

令 p 为轮廓上点的保留概率，则轮廓上的点被删除的概率为 1-p，我们在 3 个不同的 p 值下利用以上模型计算圆心的坐标。每个 p 值做了 10 次模拟（模拟程序见附录七），模拟的结果如下：

表二 稳定性检验表

p	总偏移距离	平均偏移距离	单个点最大偏移距离
0.9	1	0.02	1
0.8	5	0.1	1
0.7	13.414	0.2682	1.414

其中每个点的偏移距离即新的圆心与前面模型求解出的圆心之间的距离；总偏移距

离即 10 在一个 p 下 10 次模拟每个点偏移距离的综合；每个 p 下做 10 次模拟，每次模拟有 5 个坐标，因此每个 p 下有 50 个点，平均偏移距离既为总偏移距离除以 50。表中各数据单位为像素。

从以上数据可以看出，在轮廓发生改变后，即使在轮廓损失了近 30%的信息量时，圆心的平均偏移距离也只有 0.2682，不到一个像素，因此该求解圆心的模型具有很好的稳定性。

十、 两部固定相机相对位置的确定

为了确定两步固定相机的相对位置，需要先建立一个相机之外的世界坐标系。在这里不妨以靶标上的点 A 为原点，**AC** 的方向为 x 轴正方向，**AE** 的方向为 y 轴正方向，以 **AC** 与 **AE** 的叉积的方向为 z 轴正方向建立直角坐标系。

建立了世界坐标系后，就可以确定两个相机的光学中心 O_1、O_2 以及像平面中心 O_{r1}、O_{r2} 的坐标。利用这四个点的坐标，就可以确定两台相机的相对位置。

利用问题 1 中的模型，两个相机都可以根据各自在自己像平面上呈的像来确定靶标平面与自己的位置关系。利用这个关系就可以求出四个点在世界坐标系中的坐标。下面将给出 O_1 的世界坐标系的求法，其他三个点方法类似。

如下图所示，回到针孔相机模型的世界坐标系中，O_1 仍为坐标原点，A,B,C,D,E 各点的坐标均为已知：

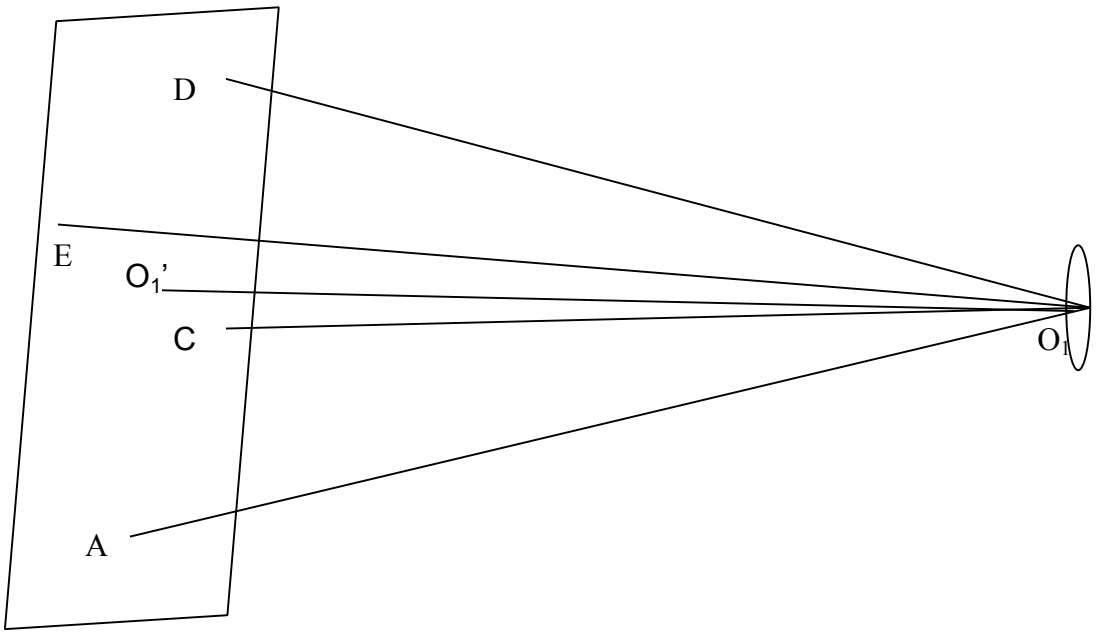

图 14 确定 O_1 点坐标示意图

如图所示过 O_1 做靶标平面的垂线 O_1O_1'，并与靶标平面相交于 O_1'。设 O_1O_1'的长度为 h，根据体积守恒，可以根据下式计算 h：

$$1/6\times|(\mathbf{O_1A}\times\mathbf{O_1C})\cdot\mathbf{O_1E}|=1/2\times h\times|\mathbf{AE}|\times|\mathbf{AC}|$$

根据 O_1O_1'的长度 h 以及 $\mathbf{O_1O_1'}$与 **AE**,**AC** 的垂直关系，求 O_1'的坐标。得到坐标后，计算 a 和 b，使得：$\mathbf{AO_1'}=a\mathbf{AC}+b\mathbf{AE}$。

最后，返回到相机之外的以 A 为原点的世界坐标系中，O_1 的坐标就可以表示为(a,b,h)。

利用上述方法，得到四个点 O_1、O_2、O_{r1}、O_{r2} 在世界坐标系中的坐标，就可以得到两部相机的位置关系。

十一、 模型扩展

以上我们建立的是线性的模型，然而实际上，由于镜头畸变、大气折射等因素的影响，实际成像过程并不满足共线条件，而是一种非线性几何关系。因此可以考虑建立一个多项式模型来综合考虑以上因素的影响。以下我们建立一个有理函数模型：

在有理函数模型中，像点坐标(r,c)表达为以相应地面点坐标(X,Y,Z)为自变量的多项式的比值，基本方程为：

$$\begin{cases} r_n = \dfrac{p_1(X_n, Y_n, Z_n)}{p_2(X_n, Y_n, Z_n)} \\ c_n = \dfrac{p_3(X_n, Y_n, Z_n)}{p_4(X_n, Y_n, Z_n)} \end{cases} \qquad (11)$$

式中 r_n 和 c_n 分别是标准化的像素坐标，X_n，Y_n 和 Z_n 分别是标准化的地面坐标。所谓标准化是指 2 个像点坐标和 3 个地面点坐标分别平移和缩放使它们的值落在[-1.0,1.0]中。一般地，多项式次数最高取 3，地面点用三维坐标表示，则每个多项式有以下形式：

$$p = \sum_{i=0}^{m_1}\sum_{j=0}^{m_2}\sum_{k=0}^{m_3} a_{ijk} * X^i * Y^j * Z^k \qquad (12)$$

其中 a_{ijk} 为有理函数多项式系数，$0 \le m_1, m_2, m_3 \le 3$，$(m_1 + m_2 + m_3) \le 3$。当函数 $p_2 = p_4 = 1$ 时，上式变为一般多项式。光学投影引起的畸变表示为一阶多项式，而镜头畸变、大气折射和地面曲率等变形可由二阶多项式来描述，高阶部分和其它未知畸变可用三阶多项式来趋近。(1)式是有理函数模型的正解形式，同样通过逆变换可得到 X_n，Y_n 的反解形式。

答解有理函数模型的实质就是确定多项式的系数，而系数的数目随多项式阶数、分母多项式的不同组合而变化，共有 9 种情况，如下表：

表三 系数数目决定

多项式阶数	分母组合	多项式系数数目	至少需控制点个数
3	p2≠p4	78	39
3	p2=p4	59	30
3	p2=p4=1	40	20
3	p2≠p4	38	19
2	p2=p4	29	15
2	p2=p4=1	20	10
2	p2≠p4	14	7
1	p2=p4	11	6
1	p2=p4=1	8	4

由于(1)式为非线性方程，可对其进行线性化处理建立其误差方程：

$$\begin{cases} v_r = \dfrac{p_1(X_n, Y_n, Z_n)}{p_2(X_n, Y_n, Z_n)} - r \\ v_c = \dfrac{p_3(X_n, Y_n, Z_n)}{p_4(X_n, Y_n, Z_n)} - c \end{cases} \qquad (13)$$

然后用最小二乘法求得相应多项式系数。

解算过程如下所示：

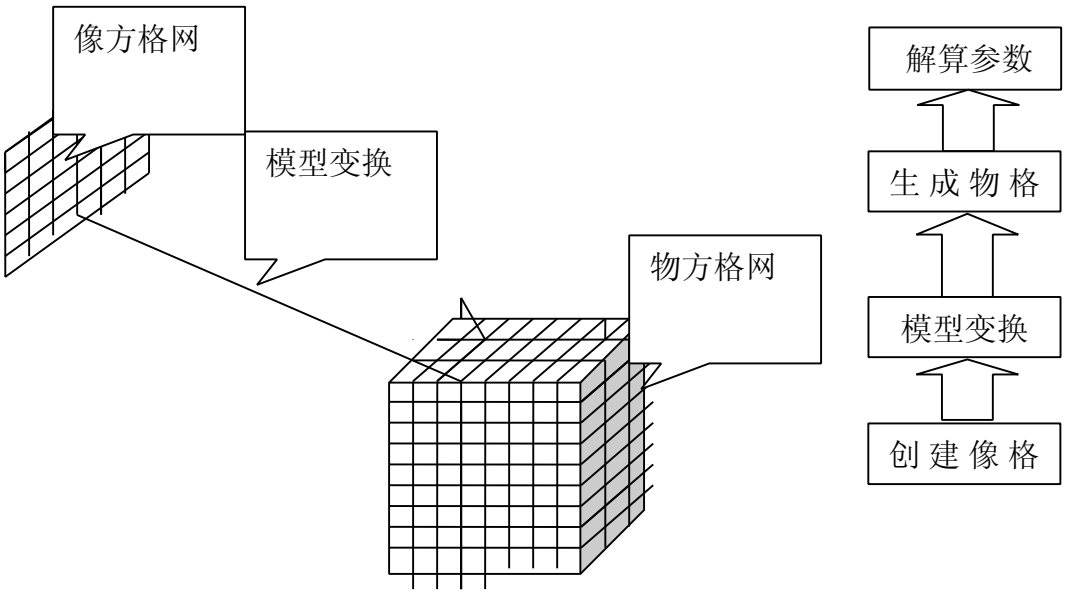

图 15 解算过程

十二、 模型评价

优点：

1.用靶标像的四条公切线的交点对应于靶标上的圆的公切线的交点，由此确定靶标平面在相机坐标中的位置，用模拟退火算法和最小二乘法求解，误差较小；

2.精度检验的模型中充分考虑了靶标像边界的信息，从整体上把握了圆的作用；

3.稳定性检验中，将靶标像上的离散点分别以三个不同的概率随机去掉部分点，其异概率性和随机性增强了检验的可靠性和说服力。

缺点：

1. 本文模型没有考虑相机拍摄时的畸变情况，使得结果存在一定的误差。

十三、 参考文献

[1]陆宏伟，基于卫星编队遥感图像的对地定位算法研究，国防科技大学，2004 年 10 月 1 日。
[2]闻仲良，《高等几何》，四川：四川大学出版社，2005 年。
[3]谢文寒，基于多像灭点进行相机标定的方法研究，武汉大学，2004 年 11 月 1 日。
[4]侯建志 战丽娜 施毅，基于 Matlab 的非线性方程组求解的方法，科技资讯，2008 年 5 月 13 日。

十四、 附录

附录一 原图像数据在 Matlab 中的处理

```
%data.m
a=imread('image3.bmp','bmp');
dat=zeros(length(a(:,1)),length(a(1,:)));
dat=dat+1;
for i=1:length(a(:,1))
    for j=1:length(a(1,:))
        if a(i,j)==15
            dat(i,j)=0;
        end
    end
end
clear a;clear i;clear j;
```

附录二 提取轮廓源程序

```
%boundary.m
%提取边界
data;
bound=zeros(length(dat(:,1)),length(dat(1,:)));
for i=1:length(dat(:,1))
    for j=1:length(dat(1,:))
        if dat(i,j)==1
            if dat(i-1,j)==1 & dat(i+1,j)==1 & dat(i,j-1)==1 & dat(i,j+1)==1
                bound(i,j)=0;
            else
                bound(i,j)=1;
            end
        else
            bound(i,j)=0;
        end
    end
end
%检测孤立点
for i=1:length(dat(:,1))
    for j=1:length(dat(1,:))
        if dat(i,j)==1
            if dat(i-1,j)==0 & dat(i+1,j)==0 & dat(i,j-1)==0 & dat(i,j+1)==0 & dat(i-
1,j+1)==0 & dat(i-1,j-1)==0 & dat(i+1,j+1)==0 &dat(i+1,j-1)==0
                bound(i,j)=0;
            end
        end
    end
end
clear i;clear j;
```

附录三 提取轮廓坐标源程序

```
%bound_zb.m
%求轮廓坐标
boundary;
zb_bound={};
h=0;a=bound;
while h==0
    m=[];
    for i=1:length(a(:,1))
        for j=1:length(a(1,:))
```

```
            if a(i,j)==1
                break;
            end
        end
        if a(i,j)==1
            break;
        end
    end
    if i==768 & j==1024
        h=1;
        break;
    else
        hh=0;
        while hh==0
            a(i,j)=0;
            m=[m;i,j];
            if a(i+1,j)==1
                i=i+1;
            elseif a(i-1,j)==1
                i=i-1;
            elseif a(i,j+1)==1
                j=j+1;
            elseif a(i,j-1)==1
                j=j-1;
            elseif a(i-1,j-1)==1
                i=i-1;j=j-1;
            elseif a(i-1,j+1)==1
                i=i-1;j=j+1;
            elseif a(i+1,j-1)==1
                i=i+1;j=j-1;
            elseif a(i+1,j+1)==1
                i=i+1;j=j+1;
            else
                hh=1;
            end
        end
    end
    for i=1:length(a(:,1))
        for j=1:length(a(1,:))
            if a(i,j)==1
                if a(i-1,j)==0 & a(i+1,j)==0 & a(i,j-1)==0 & a(i,j+1)==0 & a(i-1,j+1)==0
& a(i-1,j-1)==0 & a(i+1,j+1)==0 & a(i+1,j-1)==0
                    a(i,j)=0;
                end
```

```
                end
            end
        end
        zb_bound=[zb_bound,{m}];
end
zb_bound(2)=[];
clear a;clear h;clear i;clear j;clear hh;clear m;
```

附录四 求取切线程序

```
%qx.m
%求切线主程序
bound_zb;
cd 'qx';
sa

%isMatch.m
%判断向量是否在可行域
function f=isMatch(r,zb_y)
zb_r=zb_qx(r);
a=zb_r(:,1)*10000+zb_r(:,2);
b=zb_y(:,1)*10000+zb_y(:,2);
for i=1:length(b)
        if ~ all(a-b(i))
                f=0;return;
        end
end
f=1;return;

%mbFunction.m
%目标函数
function f=mbFunction(r)
f=abs((1/2)*det([r(1),r(2),1;r(3),r(4),1;r(7),r(8),1]))+abs((1/2)*det([r(1),r(2),1;r(5),r(6),1;r(7),r
(8),1]));
return;

%randomDisturb.m
%生成随机扰动
%n 为向量维数
%m 为最大扰动
function f=randomDisturb(n,Tm,T,Tn,L0,L1)
m=((L1-L0)/(Tn-Tm))*T+L0-Tm*((L1-L0)/(Tn-Tm));
a=round((rand(1,n)*2-1)*m);
```

```
f=a;return;

%zb_qx.m
%求切线坐标集合
function f=zb_qx(r)
jh=[];
x=1:768;
x=x';
a=[];b=[];
if r(3)~=r(1)
    y=round(((r(4)-r(2))/(r(3)-r(1)))*x+r(2)-((r(4)-r(2))/(r(3)-r(1)))*r(1));
    a=[x,y];
else
    y=1:1024;y=y';
    a=zeros(1024,1);a=a+r(1);
    a=[a,y];
end
for i=1:length(a(:,1))
    if a(i,1)>=min([r(3),r(1)]) & a(i,1)<=max([r(3),r(1)])
        b=[b;a(i,:)];
    end
end
jh=[jh;b];
y=1:1024;
y=y';
a=[];b=[];
if r(6)~=r(2)
    x=round(((r(5)-r(1))/(r(6)-r(2)))*y+r(1)-((r(5)-r(1))/(r(6)-r(2)))*r(2));
    a=[x,y];
else
    x=1:768;x=x';
    a=zeros(768,1);a=a+r(2);
    a=[x,a];
end
for i=1:length(a(:,1))
    if a(i,2)>=min([r(2),r(6)]) & a(i,2)<=max([r(2),r(6)])
        b=[b;a(i,:)];
    end
end
jh=[jh;b];
y=1:1024;
y=y';
a=[];b=[];
if r(4)~=r(8)
```

```
    x=round(((r(7)-r(3))/(r(8)-r(4)))*y+r(3)-((r(7)-r(3))/(r(8)-r(4)))*r(4));
    a=[x,y];
else
    x=1:768;x=x';
    a=zeros(768,1);a=a+r(8);
    a=[x,a];
end
for i=1:length(a(:,1))
    if a(i,2)>=min([r(8),r(4)]) & a(i,2)<=max([r(8),r(4)])
        b=[b;a(i,:)];
    end
end
jh=[jh;b];
x=1:768;
x=x';
a=[];b=[];
if r(7)~=r(5)
    y=round(((r(8)-r(6))/(r(7)-r(5)))*x+r(6)-((r(8)-r(6))/(r(7)-r(5)))*r(5));
    a=[x,y];
else
    y=1:1024;y=y';
    a=zeros(1024,1);a=a+r(7);
    a=[a,y];
end
for i=1:length(a(:,1))
    if a(i,1)>=min([r(7),r(5)]) & a(i,1)<=max([r(7),r(5)])
        b=[b;a(i,:)];
    end
end
jh=[jh;b];
f=jh;return;

%sa.m
%模拟退火算法
r=[139,256,546,235,172,690,544,617];%初始解
zb_y=[zb_bound{1};zb_bound{2};zb_bound{3};zb_bound{4};zb_bound{5}];
Tm=100;
T=Tm;%初始温度
Tn=0.01;%最低温度
E=[];%能量
dE=[];%能量差
k=0;%迭代次数
l=length(r);
L0=10;L1=1;%随机扰动范围随温度变化，最大和最小范围
```

```
best=r;zx=mbFunction(r);
while T>=Tn
    k=k+1;
    h=0;
    while h==0
        hh=0;
        while hh==0
            r1=r+randomDisturb(l,Tm,T,Tn,L0,L1);
            if isMatch(r1,zb_y)
                hh=1;
            end
        end%扰动产生一个新可行解
        E=mbFunction(r1);
        dE=E-mbFunction(r);
        if dE<=0
            r=r1;
            h=1;
        else
            p=exp(-dE/T);
            if p>=rand(1)
                r=r1;
                h=1;
            end
        end
        if mod(k,10)==0
            T=0.9*T;T
        end
        if E<zx
            zx=E;best=r;
        end
    end
end
best
mbFunction(best)
clear E;clear T;clear Tn;clear dE;clear h;clear hh;clear k;clear l;clear p;clear r1;
cd '..'
```

附录五 其圆心坐标程序

```
%qx.m
%求切线主程序
bound_zb;
%cd 'qx';
```

```
%sa;
best=[143 285 539 240 177 685 537 613];
zb0=[best(2) best(1);best(4) best(3);best(6) best(5);best(8) best(7)];
zb_x=[];
for i=1:length(zb0(:,1))
    zb_x=[zb_x;wl2xj(xs2wl(zb0(i,:)))];
end
clear zb0;

%wl2xj.m
function f=wl2xj(a)
a=[a';-1];
b=inv([-1577 0 0;0 -1577 0;0 0 1])*a;
f=[b(1),b(2),b(3)];return;

%xj2wl.m
function f=xj2wl(a)
a=[a';1];
b=[-1577 0 0 0;0 -1577 0 0;0 0 -1 0]*a;
c=[];
c=[c,b(1)/b(3)];c=[c,b(2)/b(3)];
f=c;return;

%xs2wl.m
function f=xs2wl(a)
a=[a';1];
h=inv([1 0 512.5;0 1 384.5;0 0 1])*a;
f=[h(1) h(2)];return;

%wl2xs.m
function f=wl2xs(a)
a=[a';1];
h=[1 0 512.5;0 1 384.5;0 0 1]*a;
f=[h(2) h(1)];return;

%fu.m
function f=fu(l)
c=[0.14426125554851,0.153138871274572,-1;0.17279644895371,-0.097970830691186,-1;-
0.109384908053266,0.131578947368421,-1;-0.063728598604946,-0.096702599873177,-1;];
oa=l(1)*c(1,:);
ob=l(2)*c(2,:);
oc=l(3)*c(3,:);
od=l(4)*c(4,:);
ab=ob-oa;
```

```
ac=oc-oa;
cd=od-oc;
bd=od-ob;
f(1)=dot(ab,ac);
f(2)=dot(ac,cd);
f(3)=dot(cd,bd);
f(4)=dot(ab,bd);
f(5)=norm(ab)-468.72;
f(6)=norm(ac)-468.72;
f(7)=norm(cd)-468.72;
f(8)=norm(bd)-468.72;
return;

%qj.m
res_l=fsolve(@fu,[1500 1500 1500 1500])
ca=[0.14426125554851,0.153138871274572,-1;0.17279644895371,-0.097970830691186,-1;-
0.109384908053266,0.131578947368421,-1;-0.063728598604946,-0.096702599873177,-1;];
oa=res_l(1)*ca(1,:);
ob=res_l(2)*ca(2,:);
oc=res_l(3)*ca(3,:);
od=res_l(4)*ca(4,:);
ab=ob-oa;
ac=oc-oa;
cd=od-oc;
bd=od-ob;
wc_cj=abs(dot(ab,ac));
wc_cj=wc_cj+abs(dot(ac,cd));
wc_cj=wc_cj+abs(dot(cd,bd));
wc_cj=(wc_cj+abs(dot(ab,bd)))/4
wc_cd=abs(norm(ab)-468.72);
wc_cd=wc_cd+abs(norm(ac)-468.72);
wc_cd=wc_cd+abs(norm(cd)-468.72);
wc_cd=(wc_cd+abs(norm(bd)-468.72))/4
[oa;ob;oc;od]
a3a=(12/124)*ac+(12/124)*ab;
a3b=(42/124)*ac+(12/124)*ab;
a3c=(112/124)*ac+(12/124)*ab;
a3d=(112/124)*ac+(112/124)*ab;
a3e=(12/124)*ac+(112/124)*ab;
oA=oa+a3a;
oB=oa+a3b;
oC=oa+a3c;
oD=oa+a3d;
oE=oa+a3e;
```

```
A=xj2wl(oA);B=xj2wl(oB);C=xj2wl(oC);D=xj2wl(oD);E=xj2wl(oE);
[A;B;C;D;E]
[round(wl2xs(A));round(wl2xs(B));round(wl2xs(C));round(wl2xs(D));round(wl2xs(E))]
```

附录六 计算精度误差程序

```
%jdwc.m
%计算精度误差
bound_zb;
qj;
wc=[];
jl=[];
jlj=[];
nzb=[];
yx=[oA;oB;oC;oE;oD];
for i=1:5
    for j=1:length(zb_bound{i}(:,1))
        h=-oa*inv([xs2wl([zb_bound{i}(j,2),zb_bound{i}(j,1)]),1577;ab;ac]);
        nzb=-h(1)*[xs2wl([zb_bound{i}(j,2),zb_bound{i}(j,1)]),1577];
        jl=norm(nzb-yx(i,:));
        jlj=[jlj;jl];
    end
    wc=[wc;mean(jlj),std(jlj)];
    jlj=[];
end
wc
wc(:,1)-12*3.78
wc(:,2)./wc(:,1)
```

附录七 稳定性模拟程序

注：以下程序是在前面附录的基础上修改或增加而来的，为避免重复，这里只有修改过的程序，其他相关程序请参照前面附录

```
%wdx.m
bound_zb;
dqp=0.3;
wdx_zb=zb_bound;
total_result={};
for cs=1:10
    zb_bound=wdx_zb;
```

```
    for i=1:5
        for j=length(zb_bound{i}(:,1)):-1:1
            ranum=rand(1);
            if ranum<dqp
                zb_bound{i}(j,:)=[];
            end
        end
    end
    qj;
    total_result=[total_result,{result}];
end

%qj.m
qx;
res_l=fsolve(@fu,[1500 1500 1500 1500]);
load -ascii c.txt;
ca=c;
oa=res_l(1)*ca(1,:);
ob=res_l(2)*ca(2,:);
oc=res_l(3)*ca(3,:);
od=res_l(4)*ca(4,:);
ab=ob-oa;
ac=oc-oa;
cd=od-oc;
bd=od-ob;
a3a=(12/124)*ac+(12/124)*ab;
a3b=(42/124)*ac+(12/124)*ab;
a3c=(112/124)*ac+(12/124)*ab;
a3d=(112/124)*ac+(112/124)*ab;
a3e=(12/124)*ac+(112/124)*ab;
oA=oa+a3a;
oB=oa+a3b;
oC=oa+a3c;
oD=oa+a3d;
oE=oa+a3e;
A=xj2wl(oA);B=xj2wl(oB);C=xj2wl(oC);D=xj2wl(oD);E=xj2wl(oE);
[A;B;C;D;E];
result=[round(wl2xs(A));round(wl2xs(B));round(wl2xs(C));round(wl2xs(D));round(wl2xs(E))
]

%qx.m
%求切线主程序
cd 'qx';
sa;
```

```
%best=[143     285     539     240     177     685     537     613];
zb0=[best(2) best(1);best(4) best(3);best(6) best(5);best(8) best(7)];
zb_x=[];
for i=1:length(zb0(:,1))
      zb_x=[zb_x;wl2xj(xs2wl(zb0(i,:)))];
end
save c.txt zb_x -ascii
clear zb0;

%fu.m
function f=fu(l)
load -ascii c.txt;
oa=l(1)*c(1,:);
ob=l(2)*c(2,:);
oc=l(3)*c(3,:);
od=l(4)*c(4,:);
ab=ob-oa;
ac=oc-oa;
cd=od-oc;
bd=od-ob;
f(1)=dot(ab,ac);
f(2)=dot(ac,cd);
f(3)=dot(cd,bd);
f(4)=dot(ab,bd);
f(5)=norm(ab)-468.72;
f(6)=norm(ac)-468.72;
f(7)=norm(cd)-468.72;
f(8)=norm(bd)-468.72;
return;

%sa.m
%模拟退火算法
r=[143,285,539,240,177,685,537,613];%初始解
zb_y=[zb_bound{1};zb_bound{2};zb_bound{3};zb_bound{4};zb_bound{5}];
Tm=5;
T=Tm;%初始温度
Tn=1;%最低温度
E=[];%能量
dE=[];%能量差
k=0;%迭代次数
l=length(r);
L0=3;L1=1;%随机扰动范围随温度变化，最大和最小范围
best=r;zx=mbFunction(r);
while T>=Tn
```

```
    k=k+1;
    h=0;
    while h==0
        hh=0;
        while hh==0
            r1=r+randomDisturb(l,Tm,T,Tn,L0,L1);
            if isMatch(r1,zb_y)
                hh=1;
            end
        end%扰动产生一个新可行解
        E=mbFunction(r1);
        dE=E-mbFunction(r);
        if dE<=0
            r=r1;
            h=1;
        else
            p=exp(-dE/T);
            if p>=rand(1)
                r=r1;
                h=1;
            end
            h=1;
        end
        if mod(k,10)==0
            T=0.9*T;T;
        end
        if E<zx
            zx=E;best=r;
        end
    end
end
best
mbFunction(best)
clear E;clear T;clear Tn;clear dE;clear h;clear hh;clear k;clear l;clear p;clear r1;clear L0;clear
L1;clear Tm;clear r;clear zb_y;clear zx;
cd '..'
```