数码相机定位

摘　要

本文对数码相机定位问题，建立了基于椭圆方程拟合的圆心像坐标定位模型和基于射影变换几何不变性的圆心像坐标定位模型，解决了圆心像坐标定位问题，并设计了一种方法检验模型的稳定性和精度；建立了靶标平面到成像平面的欧几里得空间映射变换模型，解决了确定两部固定相机相对位置的问题，并提出了一种检验方法。

针对问题一，首先建立了基于椭圆方程拟合的圆心像坐标定位模型，解决了圆心像坐标定位问题。首先将所得图像用高斯滤波法去噪，并用 canny 算法提取椭圆边缘，并得到边缘坐标；其次确定图像坐标系和相继坐标系；然后根据得到的边缘坐标，用最小二乘法拟合椭圆曲线方程；最后设计算法确定椭圆中心，即为圆心像坐标。最终得到圆心像坐标分别为（-50.13,51.22）、(-23.80,49.47)、（33.52,45.64）、（18.52,-31.37）、(-60.25,-31.34)。

为了进一步提高模型的精度，我们建立了射影变换几何不变性的圆心像坐标定位模型。首先分析靶标圆与其像之间的投影关系；其次设计算法求椭圆的公切线；然后建立射影变换几何不变性的圆心像坐标定位模型求解圆心的像坐标；最后设计了一种方法检验模型的精度和稳定性，并与方法一的求解结果进行对比。最终得到圆心像坐标分别为(-50.960, 51.247) 、(-24.127, 49.220) 、(33.832, 44.849) 、(18.825, -31.797) 、(-61.174, -31.560) 。模型一的精确度为 0.19，模型二的精确度为 0.056。此模型稳定，且求解精度更优。

针对问题四，建立了靶标平面到成像平面的欧几里得空间映射变换模型，解决了确定两部固定相机相对位置的问题。首先建立靶标平面到成像平面的欧几里得空间映射变换模型，并得到靶标平面到像平面的单应性矩阵来确定相机内外参数；其次重新建立三维坐标系，并利用最小二乘法计算三维坐标；然后根据求得数据推倒两个相机的相对位置；最后对模型提出了检验方法。

关键词：最小二乘法 高斯滤波 canny 算法 射影变换 欧几里得空间映射变换

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，如图 1 所示，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。

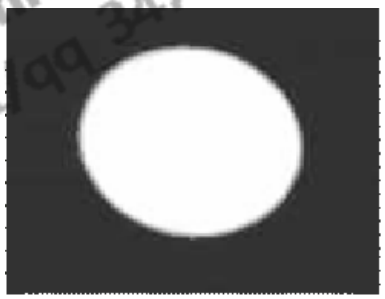

图 1 靶标上圆的像

有人设计靶标如下，取 1 个边长为 100mm 的正方形，分别以四个顶点（对应为 A、C、D、E）为圆心，12mm 为半径作圆。以 AC 边上距离 A 点 30mm 处的 B 为圆心，12mm 为半径作圆，如图 2 所示。

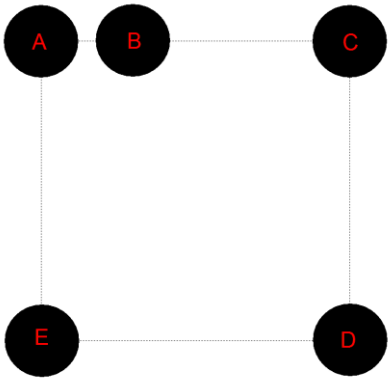

图 2 靶标示意图

用一位置固定的数码相机摄得其像，如图 3 所示。

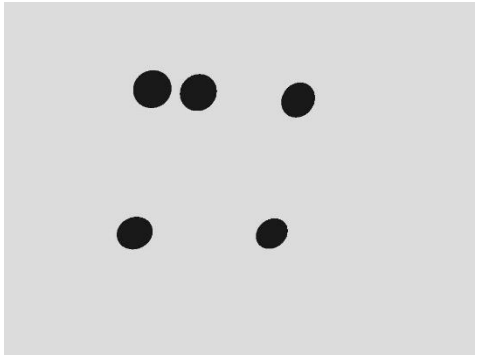

图 3 靶标的像

请你们：

(1)建立数学模型和算法以确定靶标上圆的圆心在该相机像平面的像坐标，这里坐标系原点取在该相机的光学中心，x-y 平面平行于像平面；

(2)对由图 2、图 3 分别给出的靶标及其像，计算靶标上圆的圆心在像平面上的像坐标，该相机的像距（即光学中心到像平面的距离）是 1577 个像素单位(1 毫米约为 3.78 个像素单位)，相机分辨率为 1024×768；

(3)设计一种方法检验你们的模型，并对方法的精度和稳定性进行讨论；

(4)建立用此靶标给出两部固定相机相对位置的数学模型和方法。

二、问题分析

2.1 模型一的分析

建立模型以确定靶标圆的像坐标，通常的做法是根据小孔成像原理进行研究。基于此原理，我们在模型建立之前需要对模型进行相应的准备。首先，对获取的图像需要进行处理。为了使得所给的图像拥有较高的精度且便于后续的研究，一般情况下选择对图像进行高斯去噪随后提取边缘，提取边缘常见的算法有 canny 算法、小波分析法[1]。而由于小波分析法多用于自然生物领域[2]，在这里我们选择 canny 算法。图像处理后，为了便于分析我们要建立合适的坐标系。因为有物平面和像平面的存在，我们需要分别建立图像坐标系和相机坐标系。接着，对像进行标定。因为像平面上所成的像大多为椭圆，我们利用最小二乘法拟合椭圆的曲线方程，并确定椭圆的中心。

2.2 模型二的分析

由于在测量中噪声的存在[3]，模型一中采用的最小二乘的方法并不能处理射影变换的情况，因此，我们需要对模型一进行优化以计算靶标上圆的圆心在像平面上的像坐标。对于该种情况，我们选择基于射影变换几何不变性的理论对圆心像坐标进行求解。要想利用射影不变性，其前提条件是存在一些不随视点变化的和描述子[4]，即我们需要知道靶标与像之间的定性投影关系。其次，通过拟合的椭圆曲线求得公切线。最后，通过求得的公切线确定圆心的像坐标。

2.3 问题三的分析

为了说明模型的可行性，我们从模型的精度和稳定性对模型进行检验。首先，我们需要计算像点的坐标，利用像点的坐标判断点与其成像是否共线，通过利用靶标到像平面所成像的交比不变性判断模型的稳定性[5]。其次，根据摄影变换的线性不变性，所成像的中心理论上应在一条直线上。但由于偏差的存在，像的中心应该不都在一条直线上。因此，我们计算某一个像中心到另外像中心连线的距

离，通过距离的长短来判断模型的精度，得出模型一和模型二的可行性。

2.4 问题四的分析

要想通过靶标确定固定相机的相对位置，我们首先要对相机获取的图像进行预处理。其次，通过相机的内外参数对数码相机进行标定，一般这种情况我们选择利用小孔成像原理，建立靶标平面到像面单应性矩阵[6]，倒推得到相机的内外参数。再次，由于左右观测点在观看时会产生一定的偏差，这时我们选择对观测点进行三维重建[7]，利用最小二乘法确定目标点的唯一三维坐标。最后，根据其坐标求得两个相机的相对位置。

三、基本假设

1、通常在视觉测量中，摄像机遵循小孔成像；
2、数码相机的焦距保持不变；
3、数码相机的机械误差忽略不计，如像素定位不准、质量问题等；
4、不考虑图像的噪声及其他条件的影响；
5、天气等客观因素及其他人为因素的影响忽略不计。

四、符号说明

S	像素点矩阵
S_{ij}	各像素点的像素值
U	椭圆内部点的集合
A′	A 点对应像点
B′	B 点对应像点
C′	C 点对应像点
D′	D 点对应像点
cross	交比
d	O_2 到圆心连线的距离
S	模型精确度
$x^{'}$	x 的齐次坐标
λ	比例因子
f_x	摄像机焦距
f_y	摄像机焦距
s	相平面坐标轴的倾斜系数
u_0	摄像机镜头光学中心在像面上的投影
v_0	摄像机镜头光学中心在像面上的投影
R	旋转矩阵
T	平移向量
H	靶标平面与像平面间的单应性矩阵

M	摄像头的投影矩阵
Z_c	摄像机坐标系下的 Z 坐标
z	主光轴与物平面的夹角
L_i	相机到物平面的主光轴长度
l_i	相机光学中心到像平面的距离

五、模型的建立与求解

5.1 模型的准备

5.1.1 图像预处理

第一步，读取图像信息，并将图像二值化[8]。图像二值化就是将图像上的像素点的灰度值设置为 0 或 255，也就是将整个图像呈现出明显的只有黑和白的视觉效果。

第二步，生成 1024*768 的像素点 S 矩阵，并用高斯滤波法[9]去噪。高斯滤波是一种线性平滑滤波，适用于消除高斯噪声，广泛应用于图像处理的减噪过程。通俗的说，高斯滤波就是对整个图像进行加权平均的过程，每一个像素点的值，都由其本身和邻域内的其他像素值经过加权平均后得到。将题目中所给图像经过高斯滤波法处理后的结果如图 1 所示：

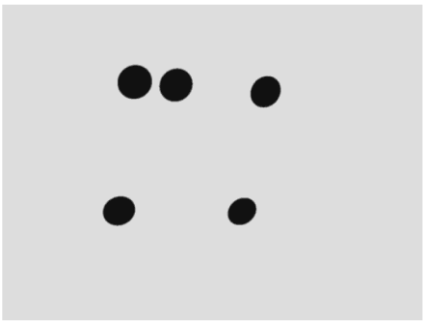

图 1 高斯去噪后的图像

第三步，用 canny 算法提取边缘[10]。这里我们对 canny 算法的步骤做如下解释：

(1) 降噪

由于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的工作，所以我们对原始数据与高斯 mask 作卷积，使得得到的图像与原始图像相比有轻微的模糊，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) 寻找图像中的亮度梯度

图像中的边缘可能会指向不同的方向，所以 canny 算法使用 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。对于每个点我们都标识在这个点上的最大值以及生成的边缘的方向。这样我

们就从原始图像生成了图像中每个点亮度梯度图以及亮度梯度的方向。

(3) 在图像中跟踪边缘

较高的亮度梯度很有可能是边缘，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。即从一个较大的阈值开始，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。在跟踪时，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回到起点。一旦过程完成，我们就得到了一个二值图像，每个点表示是否为边缘点。

第四步，得到边缘坐标。

Canny 算子分割结果如图 2 所示：

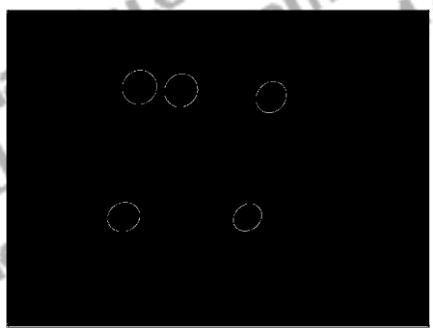

图 2　canny 算子分割后的图像

5.1.2 坐标系的建立

1、图像坐标系[11]的确定

由于每幅数字图像在计算机内的存储为 M × N数组， M 行 N 列的图像中的每一个元素的数值即为图像点的亮度。如图 3 所示，在图像上定义直角坐标系 uO_0v，每一像素的坐标(u,v)分别是该像素在数组中的列数与行数。所以，(u,v)是以像素为单位的图像坐标系。

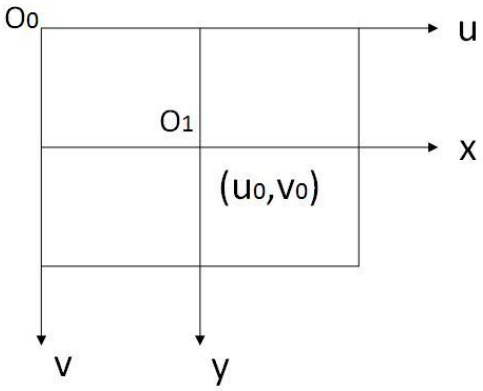

图 3 图像坐标系

由于(u,v)只表示像素位于数组中的列数与行数，并没有用物理单位表示出该像素中的位置，因而，需要在建立以长度（毫米）单位表示的图像坐标系。该坐标系以图像内某一点O_1为原点，x 轴和 y 轴分别与 u,v 轴平行。在 x、y 坐标系中，原点O_1定义在摄像机光轴与图像平面的交点，该点一般位于图像中心处，但由于摄像机制作的原因，也会有些偏离，若O_1在 u,v 坐标系中的坐标为u_0, v_0，

每一个像素在 x 轴与 y 轴方向上的物理尺寸为 dx,dy，则图像中任意一个像素在两个坐标系下的坐标有如下关系：

$$u = \frac{x}{dx} + u_0$$

$$v = \frac{y}{dy} + v_0$$

为了方便使用，可以用齐次坐标与矩阵形式将上式表示为：

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

逆关系可写成：

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} dx & 0 & -u_0 dx \\ 0 & dy & -v_0 dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

2、相机坐标系[12]的确定

过图像的中心点作像平面的垂线，在垂线上取一点 O 作为相机的光心，使 O 点到像平面的距离为 l，即相机的像距，以相机的光轴为 Z 轴，X 轴和 Y 轴分别平行与图像的 x 轴和 y 轴，由点 O 分别与 X、Y、Z 轴形成的直角坐标系称为相机坐标系，如图 4 所示：

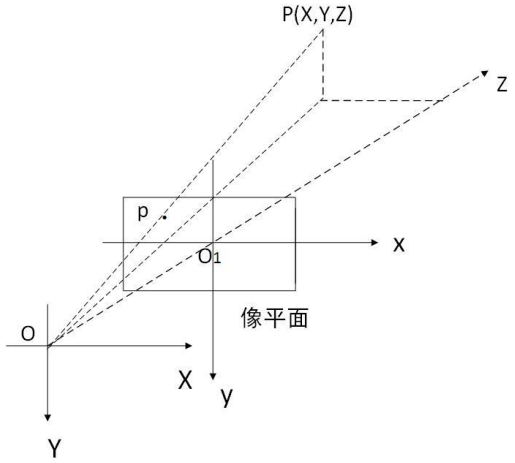

图 4 相机坐标系

5.2 模型一的建立与求解

光学测量领域，圆形是极为常见的一种图形特征。对比其他形状特征，圆形识别成功率高、噪声抑制性强且对分割阈值不敏感，在实际视觉系统中承担着重要作用[13]。但经透视投影变换，圆中心的真实投影点一般并不是投影轮廓的几何中心。因此，准确计算圆心的真实投影点有着重要的意义。

5.2.1 模型的建立

为了确定靶标上圆的圆心在相机像平面的像坐标，就要对像进行标定。在标定的过程中，通常以圆作为标志，并以其中心——圆心作为标定点。求出圆心在

图像平面的像坐标后 ,将几对圆心坐标和对应的像点坐标代入物点和像点的坐标变换公式, 摄像机参数标定即可实现。

1、最小二乘法拟合椭圆曲线方程[14]

经过图像处理，我们首先将获得椭圆边界点的像素坐标利用公式转换为物理坐标,可在像平面上拟合出椭圆曲线方程,关于该问题的文献中已有大量现成算法。本文采用基于代数距离的最小二乘法来拟合椭圆,基本算法如下:设椭圆的曲线方程为：

$$F(x,y) = x^2 + axy + by^2 + cx + dy + e = 0$$

然后利用最小二乘法代入确定系数，即通过最小化误差的平方和寻找数据的最佳函数匹配，简便的求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小，适用于曲线拟合。

最后得到椭圆方程。

2、确定椭圆的中心

为了确定椭圆的中心，我们规定其算法步骤如下：

步骤一：确定某个椭圆内部点的集合 U；

步骤二：将任意一椭圆内部点 o 以及椭圆上一点 a 连线，连线与椭圆的另一交点为 b；

步骤三：若 oa=ob，则 o 为椭圆的中心；否则返回步骤二；

步骤四：若所有椭圆中心确定，则结束；否则返回步骤一。

5.2.2 模型的求解

通过最小二乘法拟合得到 5 个椭圆的曲线方程分别为：

$$x^2 - 2.0508xy + 1.1331y^2 - 24.4951x - 28.4474y + 7304 = 0$$
$$x^2 - 3.1396xy + 2.2465y^2 - 25.7356x - 17.5923y + 7303.7 = 0$$
$$x^2 + -3.0242xy + 2.1935y^2 - 56.2710x + 20.2387y + 7303.6 = 0$$
$$x^2 - 2.3241xy + 1.2907y^2 - 62.1475x + 27.1826y + 7303.7 = 0$$
$$x^2 - 1.9728xy + 0.8818y^2 - 46.3128x + 10.0983y + 7303.9 = 0$$

所得椭圆的中心如表 1 所示：

表 1 椭圆中心坐标

椭圆	x	y
A	-50.13	51.22
B	-23.80	49.47
C	33.52	45.64
D	18.52	-31.37
E	-60.25	-31.34

通过表 1 我们得出 5 个椭圆中心的坐标分别为：（-50.13,51.22）、(-23.80,49.47)、（33.52,45.64）、（18.52,-31.37）、(-60.25,-31.34)。

5.3 模型二的建立与求解

对于靶标圆心像点的计算，传统的方法就是对圆在像平面上的像(椭圆) 进行图像处理，得到椭圆边缘像素点的离散数据。由于噪声的存在，要用这些离散数据进行椭圆的拟合，用得到的椭圆的形心作为靶标圆心像点。模型一中采用最小二乘法进行椭圆拟合，虽然便于计算，但在射影变换下,圆心的像一般不是椭圆

的中心点[15]，这种处理方法存在较大的误差。因此，我们对模型一进行改进，基于射影变换几何不变性的理论[16]，对圆心像坐标进行求解。同时,利用射影变换保持点的共线及交比不变[17]的性质,对该方法的精度和稳定性给出一种验证模型。

5.3.1 模型的建立

1、分析靶标圆与其像之间的投影关系

针孔成像模型符合透视投影变换，而高等画法几何学中透视投影变换有以下性质：点一定与点对应，且是一一对应的；直线的像仍为直线。由此可以推导出靶标圆与其像之间的性质：两圆的公切线经针孔模型所 成的像为对应两椭圆的公切线；过像平面上任一个椭圆与任意不同的两个椭圆分别作两条外公切线，则两 对切点所确定的两条直线的交点是靶标圆心的像点。如图 3 所示，A'_1、A'_2、B'_1、B'_2 分别为 A_1、A_2、B_1、B_2 的像，则直线 $A'_1A'_2$ 与 $B'_1B'_2$ 为靶标上直线 A_1A_2 与 B_1B_2 的像。又因为点 M 必为点 O′对应的像点，由圆形靶标的几何性质可知，点 O′是圆 O′的圆心，所以 M 是靶标圆心 O′的像点。如图 5 所示：

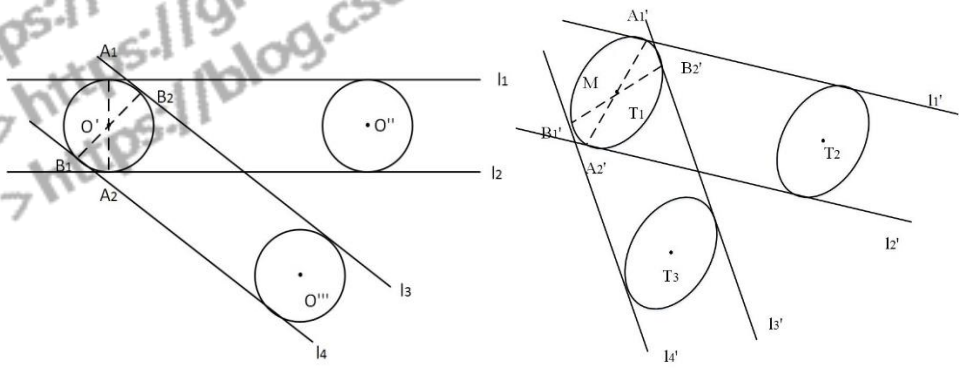

图 5 靶标与其投影示意图

接着，我们引入两个相关定义：

① 射影变换保持点的共线及交比不变性质：

若 A,B,C,D 四点共线,则经射影变换后其对应的像点 A′,B′,C′,D′仍然共线,且交比保持不变。

② 射影变换保持直线与圆的相切性质：

设直线 AB 外切于圆 O,在此射影变换作用下,直线 AB 的像 A′B′与圆 O 的像 O′相切关系仍然保持。

基于上述定义，我们建立基于射影变换几何不变性的圆心像坐标定位模型。

2、基于射影变换几何不变性的圆心像坐标定位模型

根据圆的透视投影性质可知, 圆的图像为椭圆,本文首先采用最小二乘法对圆的像椭圆曲线进行拟合,然后利用圆的公切线像的连线交点求解出圆心的像坐标。步骤如下：

① 求椭圆的公切线

切线表示线与椭圆只有一个交点，若有两个椭圆，则可以确定最多四条切线。利用这一原理，使用像平面上 A′,C′,D′, E′四个近似椭圆确定其外围四条切线。这四条切线组成一个四边形，四个近似椭圆均切于四边形某一条边。其中，四边形的顶点位置正是我们所求的，而求得顶点的关键是求得四条切线，求得切线的关键是找到切点，两点连线确定一条切线。

② 利用公切线求解圆心的像坐标

得到圆的像——椭圆曲线方程后,进一步可求出圆心的像坐标,我们给出利用公切线求解圆心的像坐标方法,基本原理如下:

设靶标上有 3 个圆 A、B、C,用一部数码相机摄得其像,如图 6 和图 7 所示。以求圆心 C 点的像坐标为例说明方法:如图 6 所示,作两圆 B,C 的外公切线,C_1, C_2 分别为其在圆 C 上的两切点,类似地,作两圆 C,A 的外公切线,C3,C4 为其在圆 C 上的两切点,由几何关系知:两直线 C_1C_2 和 C_3C_4 必过圆心,根据射影变换保持点的共线性及相切关系仍然保持知:圆心 C 的像 C′(如图 7)既在直线 $C'_1C'_2$ 上,又在直线 $C'_3C'_4$ 上,这里 $C'_i(i=1,2,3,4)$分别为 $C_i(i=1,2,3,4)$的像,故可以通过在像平面的椭圆曲线上求外公切线切点的方法得到圆心的像坐标。

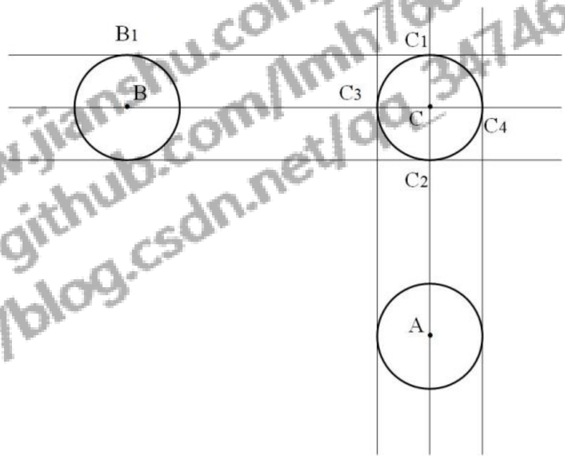

图 6 圆及其外公切线示意图

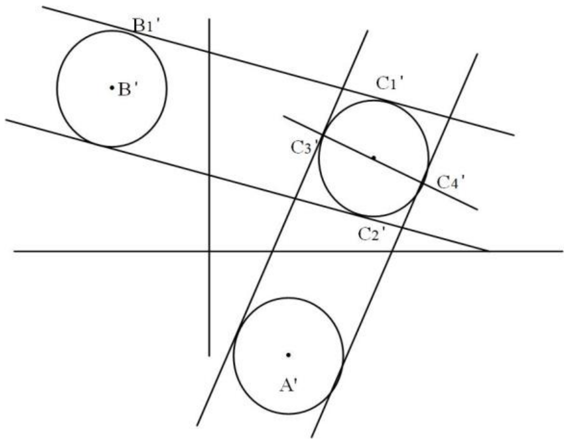

图 7 圆及其外公切线的像的示意图

具体求解步骤如下:

(1)求两椭圆外公切线的切点坐标

由于 $B'_1C'_1$ 为椭圆 B′,C′的外公切线,可知椭圆 B′,C′方程分别在切点 B'_1、C'_1 的导数和直线 $B'_1C'_1$ 斜率相等,同时切点 B'_1、C'_1 坐标分别满足椭圆 B′,C′的方程,通过对切点坐标满足的方程联立求解即可求出切点 B'_1、C'_1、C'_2 的坐标;类似地,可求出切点 C'_3、C'_4 的坐标。

(2)求直线 $C'_1C'_2$ 与 $C'_3C'_4$ 的交点,即为圆心 C 的像 C′的物理坐标。

5.3.2 模型的求解

通过以上模型进行求解，得到 5 个圆心的像坐标如表 2 所示：

表 2 圆心的像坐标

椭圆	x	y
A	-50.960	51.247
B	-24.127	49.220
C	33.832	44.849
D	18.825	-31.797
E	-61.174	-31.560

通过表 2，我们得到 5 个圆心的像坐标分别为(-50.960, 51.247)、(-24.127, 49.220)、(33.832, 44.849)、(18.825, -31.797)、(-61.174, -31.560)。

5.3.3 模型的检验

为了说明上述模型的可行性，我们设计方法对模型进行检验。

1、稳定性的检验

通过上述设计的靶标，计算得到：

$$\text{cross}(A,B;C,M_\infty)=\frac{10}{7}$$

接着，我们计算 A′, B′,C′,M′的交比，其步骤如下：

(1) 验证 A′, B′,C′共线

通过计算，我们得到：

$$\overrightarrow{A'B'}=\{26.833,-20.27\}$$

$$\overrightarrow{A'C'}=\{84.792,-6.398\}$$

由此我们发现：

$$\overrightarrow{A'C'}=\lambda\overrightarrow{A'B'},\lambda\approx 3.16$$

所以，A′, B′,C′共线。

(2) 计算 M′坐标

因为 AB//ED，所以 AC，ED 的消影点均为 M′，即 M′既在直线$A'C'$上，又在直线$E'D'$上，其中 D′为圆心 E，D 的像点。

直线$A'C',E'D'$的方程分别为：

$$y-44.849=\frac{(44.849-51.247)}{(33.832+50.960)}(x-33.832)$$

$$y+31.797=\frac{(-31.797+31.560)}{(18.825+61.174)}(x-18.825)$$

将两式联立，解得 M′坐标为(1092.3199,-34.9713)。

(3) 求 A′, B′,C′,M′四点的交比

通过计算可以得到：

$$\text{cross}(A',B';C',M')=1.4288\approx\frac{10}{7}$$

从检验的结果来看，A,B,C,M_∞的像点A',B',C',M'仍然共线，且交比保持不变。由此可见，上述模型中求圆心坐标的方法具有非常高的精确性与稳定性。

2、精度的讨论

根据摄影变换的线性不变性，O_1、O_3 应该共线，因此我们根据 O_1、O_3 确定一条直线，计算 O_2 到该直线的距离，如图 8 所示。偏差越大表明该模型不精确。

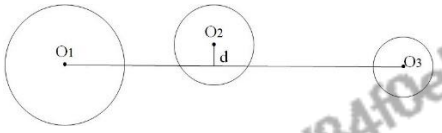

图 8 对精度的讨论

根据题目条件我们只能根据 d 判断模型的精确度。若要提高此方法的精确度，可在 AE、BD、CD 边上增加一个靶标，然后求得D_i平方和的平均值的即为该模型的精确度 S，即：

$$S = \frac{\sum_{i=1}^{n} {D_i}^2}{n}$$

由此方法计算出模型一的精确度为 0.19，模型二的精确度为 0.056.由此可见，模型二精度更高。

5.4 问题四的模型建立与求解

要想利用此靶标给出固定相机的相对位置，我们首先要对摄像机的内外参数进行标定。根据比例因子，得到像面单应性矩阵[18]并求得矩阵的线性方程，联立方程倒推求得内外参数。其次，利用三维重建找到目标点的三维位置。接着，利用投影矩阵和最小二乘法，计算该点的三维坐标。最后，确定两个相机的相对位置。

5.4.1 模型的建立

建立模型的步骤如图 9 所示：

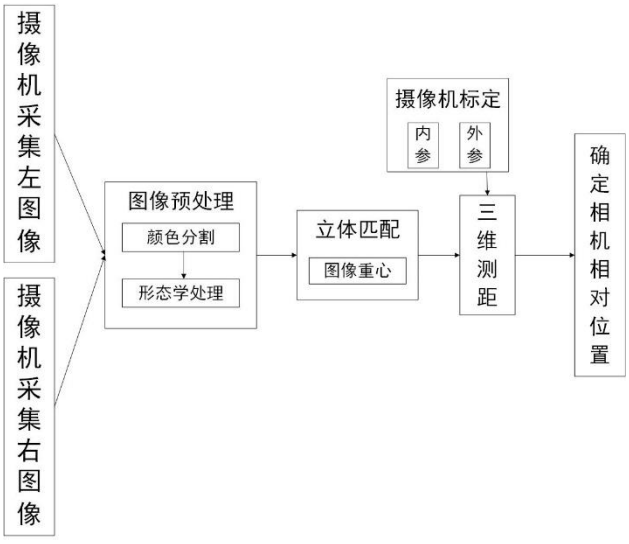

图 9 模型建立步骤

1、标定——确定相机内外参数

摄像机标定是进行目标定位的必要过程，通过摄像机标定可以得到摄像机的内外参数。常用标定方法有：传统标定方法、自标定方法等[19]。我们采用传统标定方法对摄像机进行标定。在传统标定方法中，首先摄像机的前方

放置标定参照物，由摄像机获取标定物的图像。通过图像处理的方法获取标定物上的特征点，同时对标定物特征点的世界坐标进行精确测定。最后利用一系列标定点在图像面上成像点的像素坐标与其世界坐标之间的对应关系，借助非线性优化的方法，计算出摄像机数学模型中包含的内外参数。

(1) 靶标平面到成像平面的欧几里得空间映射变换[20]模型

通常在视觉测量中，摄像机遵循小孔成像透视变换模型。假设空间任意一点$x = [X, Y, Z]^T$,对应相点$m = [u, v]^T$则其齐次坐标分别为$x' = [X, Y, Z, 1]^T$和$m' = [u, v, 1]^T$。根据透视变换模型，空间点 x 及其像点 m 应满足：

$$\lambda m' = K[R \quad T]x'$$

其中λ为非零常数比例因子；

$$K = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

f_x、f_y是摄像机焦距，s 为相平面坐标轴的倾斜系数（取决于像面坐标系 X、Y 轴的夹角），u_0、v_0为摄像机镜头光学中心在像面上的投影（称为像面中心）。上述参数属于摄像机的内部参数。

R 为3×3旋转矩阵，T 为3×1平移向量，表示了世界坐标系到摄像机坐标系的转换关系，为摄像机外部参数。选取世界坐标系的 X Y 平面与靶标平面重合，则在世界坐标系下平面靶标可表示为Z = 0 。若旋转矩阵 R 的第 i 列元素由r_i表示，则上式可变为：

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[r_1 \quad r_2 \quad r_3 \quad T] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K[r_1 \quad r_2 \quad T] \begin{bmatrix} X \\ Y \end{bmatrix}$$

仍以 x’表示平面靶标上任意一点 x 的齐次坐标，则$x' = [X, Y, 1]^T$。因此，我们得到：

$$\lambda \tilde{m} = H \tilde{x}$$

其中，$H = \frac{1}{\lambda} K[r_1 \quad r_2 \quad T] = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$，为靶标平面与像平面间的单应性矩阵，表示了靶标平面与像平面之间的二维映射关系。

由于存在比例因子λ，因此靶标平面与像平面间的映射变换为一组单应性矩阵。通过 4 个或 4 个以上位置已知的对应点，可实现单应性矩阵的计算。

(2) 靶标平面到像面单应性矩阵的建立

靶标平面到像平面单应性矩阵的建立，至少需要 4 个特征点的空间与图像对应位置信息。由于特征点在平面靶上通常都是按照行、列及一定间距规则排列，将其中任意一个特征点（通常选左上角特征）中心定义为靶标坐标系的原点，以特征点行排列定义为靶标坐标系的 x 轴（方向通常选由左到右），以特征点列排列定义为靶标坐标系的 y 轴（方向通常选由上到下）建立右手坐标系，在此坐标系下可以构建特征点的空间坐标。圆特征的图像位置可以

通过边缘检测、最小二乘椭圆拟合等算法进行精确定位，方格特征则可通过角点提取算法进行定位。构建单应性矩阵时，至少需要 4 组特征点。若取 4 点，则要求其中任 3 点不共线。由上述公式可知，每一组对应点可提供两个关于单应性矩阵 H 的线性方程，即：

$$\begin{cases} u(h_{31x} + h_{32}y + h_{33}) = h_{11x} + h_{12}y + h_{13} \\ v(h_{31x} + h_{32}y + h_{33}) = h_{21x} + h_{22}y + h_{23} \end{cases}$$

4 组对应构建 8 个线性方程，则单应性矩阵 H 矩阵（仅相差比例因子λ）可解。

将点坐标代入，求得矩阵，倒推得到相机确定相机内外参数[21]。

2、三维重建

立体视觉从左、右观测点观看对象时，可以看到对象的偏差。这种偏差与观测点到被观测对象的距离有关。三维空间内的点在投影图像上的位置是由三维空间中的某条直线决定的。因此，如果一个对象从其他视点观测也有同样的三维投影图像，那么被观测对象的空间坐标就可以确定，2 条视线的交点就是它的位置。这就是三维重建的过程。

空间点三维重建的基本模型如图 10 所示：

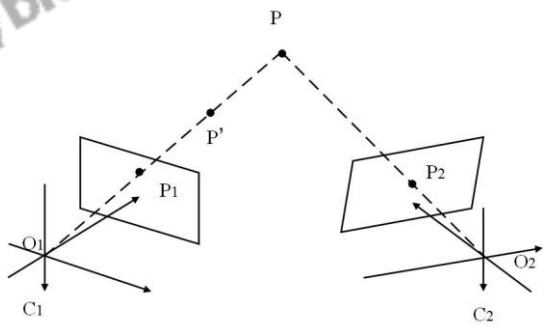

图 10 三维重建基本模型

对于空间物体表面的任意一点 P，被摄像头 C_1 拍摄下来后，在图片上位于 P_1，被摄像头 C_2 拍摄下来后，在图片上位于 P_2，但是无法仅仅通过 P_1 或者 P_2 来得出 P 的三维位置，因为在直线 O_1P 或 O_2P 上的任意一点 P'，其在图像上的位置都是 P_1 或 P_2。然而，现在利用 2 个摄像头来拍摄图片，这样就可以知道目标点位于 O_1P_1 和 O_1P_2 这 2 条直线的交点，即可以唯一确定目标点的三维位置。

3、利用最小二乘法计算三维坐标

假设 2 个摄像头的投影矩阵分别为 M_1 和 M_2，可以得到如下 2 个等式：

$$Z_{c1}\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11}^1 & m_{12}^1 & m_{13}^1 & m_{14}^1 \\ m_{21}^1 & m_{22}^1 & m_{23}^1 & m_{24}^1 \\ m_{31}^1 & m_{32}^1 & m_{33}^1 & m_{34}^1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$Z_{c2}\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11}^2 & m_{12}^2 & m_{13}^2 & m_{14}^2 \\ m_{21}^2 & m_{22}^2 & m_{23}^2 & m_{24}^2 \\ m_{31}^2 & m_{32}^2 & m_{33}^2 & m_{34}^2 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

其中，(u_1, v_1)和(u_2, v_2)分别为 P 点的成像点在 2 幅图像中的其次图像坐标。Z_{c1}和Z_{c2}分别为 P 点在 2 个摄像机坐标系下的 Z 坐标；(X,Y,Z)为 P 点在

世界坐标系下的其次坐标；利用最小二乘法，就可以求得 P 点坐标。

4、确定两个相机的相对位置

所以，当有多个点 P 时，知道其三维坐标就可求得两个相机的三维坐标，即可知道两个相机的相对位置。

5.4.2 模型的检验

现实中可以采用相机拍照得到实物照片和物体照片的相应点的对应关系，对模型的精度进行检验。通常应取得点数$n \geq 4$，通过这些点的对应关系分别求出每架相机的内外部参数，进而求得两架相机的相对位置关系。与实际情况的相对位置比较。在没有相机拍摄的情况下，为验证模型的精确度同时便于确定两相机的相对位置，我们提出一种简单验证的方法。

两个相机拍照的模型如图 11 所示：

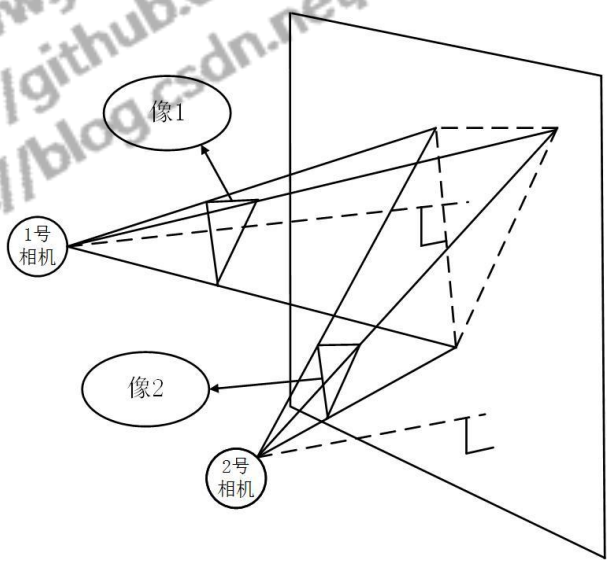

图 11 相机标定示意图

图中的平面表示物平面，平面中的三个点表示特征点，假设这三个点确定的三角形为正三角形。这三个点在相机 1 和相机 2 中的成像情况如图 11 所示。现假定相机 1 的主光轴垂直于物平面，且过正三角形的中心，那么像 1 平行于物平面。假设相机 2 的主光轴也过正三角形的中心，且成像也平行于物平面。设其与物平面的夹角为 z，并且其倾斜方向沿正三角形中心线的方向。

现给定两相机到物平面的主光轴长度分别为 L_1 和 L_2，给定相机光学中心到像平面的距离为 l_1 和 l_2。若以物平面为 xOy 面，以 1 号相机的主光轴为 z 轴，则相机 1 和 2 的相对位置可以唯一确定，同时，像点的空间坐标也唯一确定[15]。

现认为已知相机 1 和相机 2 之间的相对位置关系，根据已知的物和像的坐标运用上述模型进行求解，得到两相机的相对位置关系。以上所述验证方法基于特殊位置关系下的参数设定，旨在为两个相机设定一种一定存在的相对位置关系。比较已知和求得的相对位置关系，根据它们的差值近似得到模型的精度。

六、模型的评价与推广

6.1 模型的优点与缺点

对图像进行高斯去噪后提取边缘的预处理方法，使得所给的图像拥有较高的精度且便于后续的研究，减少了噪声对图像的干扰。接着我们基于射影变换几何不变性的理论针对模型一进行优化，解决了射影变换下圆心的像一般不是椭圆的中心点的问题，解决了最小二乘的方法不能处理的射影变换的情况，使得模型的建立更加符合实际情况。利用射影变换保持点的共线及交比不变的性质,使得该模型拥有较高的精度和较强的稳定性。确定相机的内外参数使得标定的结果更加准确。针对模型的稳定性检验，我们采取了 8 个点，体现了取点的一般性。

针对模型一，我们没有考虑相机在设计和安装过程中的畸变产生的误差。针对模型二，我们在计算过程中认为光轴通过像的中心，但是实际拍摄的过程中，由于各方面条件的影响，光轴与所成像中心存在一定的偏差。针对第四问的模型，并没有明确得出两部相机的相对位置关系。

6.2 模型的推广

本文建立的模型可以用来进行电子监控、物体的识别与定位、机器人行走引导系统、字符的目标识别、工业检测、图像处理、医疗制图等。针对问题四，我们建立的模型适用于大型物体多视角的测量的拼合、某些情况下的装配检查等。

参考文献

[1] 李嘉璐. 基于立体视觉的移动机器人定位方法研究[D].天津理工大学,2014.
[2] 谢静,庹先国,方晓. 基于边缘几何不变性的特征提取算法研究[J]. 电子技术应用,2015
[3] 陈凯,崔楠,柳有权. 基于包围盒的靶标圆像点定位算法[J]. 信息技术,2014
[4] 邹朋朋,张滋黎,王平,汪启跃,周维虎. 基于共线向量与平面单应性的双目相机标定方法[J/OL]. 光学学报,2017
[5] 程新跃.马小玉.沈玉玲.射影 Ricci 曲率及其射影不变性[J].西南大学学报（自然科学版）.2015(37)
[6] 占栋,肖建. 基于线结构光参考平面的多摄像机灵活标定方法研究[J]. 仪器仪表学报,2015,36(09):2030-2036.
[7] 陈焱明. 基于机载与车载 LiDAR 数据的建筑物模型多视三维重建研究[D].南京大学,2015.
[8] 张虎龙. 基于二维直方图和模糊熵准则的阈值化方法[J]. 计算机测量与控制,2017
[9] 彭志涛,陈风东,唐军,刘国栋,元浩宇,朱启华. 高斯差分滤波多尺度损伤提取方法[J/OL]. 强激光与粒子束,2017

[10]WORTHINGTON PL.Enhanced Canny edge detection using curvature consistency[C].Proceedings of International Conference on Pattern Recongnition,Los Alamitos,CA USA;IEEE,Computer SOC Press,2012:596-599
[11]Schaefer S,Mcphail T,Warren J.Image deformation using moving least squares[C]//Computer Graphics Proceedings,Annual Conference Series.ACM SIGGRAPH.New York:ACM Press,2013:553-540
[12]Keys R.Cubic convolution interpolation for digital image processing[J].IEEE Transactions on Imaging,1999,18(11):1049-1075
[13] 陈天飞,赵吉宾,王银灵,吴翔. 基于射影变换圆阵靶标中心像点的计算[J]. 仪器仪表学报,2015
[14] 楚圣辉,张慧萌,陈硕,孟浩,刘国忠.大场景下多目立体视觉标定方法的研究[J].现代计算机,2017
[15] 唐献献.基于双目立体视觉的目标识别与定位研究[D].哈尔滨理工大学,2017.
[16] 周前飞,刘晶红,王宣,孙明超. 航空变焦距斜视成像几何畸变的自动校正[J].光学精密工程,2015

附录

```
OriImage=imread('C:\Users\美含\Desktop\shumo\08\赛题\bbdx.jpg');%图像二值化
%预处理：高斯滤波
sigma = 1.6;
grayImg=rgb2gray(OriImage);
gausFilter = fspecial('gaussian',[5 5],sigma);
blur=imfilter(grayImg,gausFilter,'replicate');
figure(1)
imshow(blur)
%预处理：平滑滤波
I1=rgb2gray(OriImage);
k4=filter2(fspecial('average',9),I1)/255;          %进行 9*9 模板平滑滤波
%第二步：提取边缘
I2=edge(k4,'canny');  %canny 算法
figure(2)
imshow(I2) %做图
title('canny 算子分割结果');

%第三步：提取边缘坐标
[x,y]=find(I2==1);
x=x';
y=y';
x1=y;
y1=768-x;
figure(3)
scatter(x1,y1,2)

%第四步：拟合曲线
x0=zeros(1);
y0=zeros(1);
%记录上切点
x_max=[0 0 0 0 0];
y_max=[0 0 0 0 0];
%记录下切点
x_min=[720 720 720 720 720];
y_min=[720 720 720 720 720];

%计算左下角椭圆曲线方程
for i=1:length(x1)
    if (x1(i)>=200)&&(x1(i)<=400)&&(y1(i)>=200)&&(y1(i)<=400)
        if(x0(1)==0)
            x0(1)=x1(i);
```

```
            y0(1)=y1(i);
        else
            x0=[x0 x1(i)];
            y0=[y0 y1(i)];
        end
    end
end
for i=1:length(x0)
    if(y0(i)>y_max(1))
            y_max(1)=y0(i);
            x_max(1)=x0(i);
        end
        if(y0(i)<y_min(1))
            y_min(1)=y0(i);
            x_min(1)=x0(i);
        end
end
F1 =@(X1,x,y)x.*x+X1(1).*x.*y+X1(2).*y.*y+X1(3)*x+X1(4)*y+ X1(5);   % 圆锥曲线   X 为
系数
F = @(X1)F1(X1,x0,y0);
X0 =    [0.3 0.4 0.4 0.4 0.5];      % Starting guess
[Xf1,resnorm] = lsqnonlin(F,X0)       % Invoke optimizer    Xf 为拟合系数

%计算右下角椭圆曲线方程
for i=1:length(x1)
    if (x1(i)>=400)&&(x1(i)<=800)&&(y1(i)>=200)&&(y1(i)<=400)
        if(x0(1)==0)
            x0(1)=x1(i);
            y0(1)=y1(i);
        else
            x0=[x0 x1(i)];
            y0=[y0 y1(i)];
        end
    end
end
for i=1:length(x0)
    if(y0(i)>y_max(2))
            y_max(2)=y0(i);
            x_max(2)=x0(i);
        end
        if(y0(i)<y_min(2))
            y_min(2)=y0(i);
            x_min(2)=x0(i);
        end
```

```
end
F1 = @(X1,x,y)x.*x+X1(1).*x.*y+X1(2).*y.*y+X1(3)*x+X1(4)*y+ X1(5);   % 圆锥曲线   X 为系数
F = @(X1)F1(X1,x0,y0);
X0 =    [0.3 0.4 0.4 0.4 0.5]    ;       % Starting guess
[Xf2,resnorm] = lsqnonlin(F,X0)       % Invoke optimizer    Xf 为拟合系数

%计算右上角椭圆曲线方程
for i=1:length(x1)
      if (x1(i)>=600)&&(x1(i)<=800)&&(y1(i)>=500)&&(y1(i)<=700)
            if(x0(1)==0)
                  x0(1)=x1(i);
                  y0(1)=y1(i);
            else
                  x0=[x0 x1(i)];
                  y0=[y0 y1(i)];
            end
      end
end
for i=1:length(x0)
      if(y0(i)>y_max(3))
                  y_max(3)=y0(i);
                  x_max(3)=x0(i);
            end
            if(y0(i)<y_min(3))
                  y_min(3)=y0(i);
                  x_min(3)=x0(i);
            end
end
F1 = @(X1,x,y)x.*x+X1(1).*x.*y+X1(2).*y.*y+X1(3)*x+X1(4)*y+ X1(5);   % 圆锥曲线   X 为系数
F = @(X1)F1(X1,x0,y0);
X0 =    [0.3 0.4 0.4 0.4 0.5]    ;       % Starting guess
[Xf3,resnorm] = lsqnonlin(F,X0)       % Invoke optimizer    Xf 为拟合系数

%计算中间椭圆曲线方程
for i=1:length(x1)
      if (x1(i)>=400)&&(x1(i)<=600)&&(y1(i)>=500)&&(y1(i)<=700)
            if(x0(1)==0)
                  x0(1)=x1(i);
                  y0(1)=y1(i);
            else
                  x0=[x0 x1(i)];
                  y0=[y0 y1(i)];
```

```
            end
        end
    end
    for i=1:length(x0)
         if(y0(i)>y_max(4))
                y_max(4)=y0(i);
                x_max(4)=x0(i);
            end
            if(y0(i)<y_min(4))
                y_min(4)=y0(i);
                x_min(4)=x0(i);
            end
    end
    F1 = @(X1,x,y)x.*x+X1(1).*x.*y+X1(2).*y.*y+X1(3)*x+X1(4)*y+ X1(5);   % 圆锥曲线   X 为
    系数
    F = @(X1)F1(X1,x0,y0);
    X0 =    [0.3 0.4 0.4 0.4 0.5]    ;       % Starting guess
    [Xf4,resnorm] = lsqnonlin(F,X0)       % Invoke optimizer    Xf 为拟合系数

    %计算左上角椭圆曲线方程
    for i=1:length(x1)
        if (x1(i)>=200)&&(x1(i)<=400)&&(y1(i)>=500)&&(y1(i)<=700)
            if(x0(1)==0)
                x0(1)=x1(i);
                y0(1)=y1(i);
            else
                x0=[x0 x1(i)];
                y0=[y0 y1(i)];
            end
        end
    end
    for i=1:length(x0)
         if(y0(i)>y_max(5))
                y_max(5)=y0(i);
                x_max(5)=x0(i);
            end
            if(y0(i)<y_min(5))
                y_min(5)=y0(i);
                x_min(5)=x0(i);
            end
    end
    F1 = @(X1,x,y)x.*x+X1(1).*x.*y+X1(2).*y.*y+X1(3)*x+X1(4)*y+ X1(5);   % 圆锥曲线   X 为
    系数
    F = @(X1)F1(X1,x0,y0);
```

```
X0 =    [0.3 0.4 0.4 0.4 0.5]    ;       % Starting guess
[Xf5,resnorm] = lsqnonlin(F,X0)       % Invoke optimizer    Xf 为拟合系数
%{
figure(4)
Y1=ezplot('x^2-2.0508*x*y+1.1331*y^2-24.4951*x -28.4474*y+7304=0',[0,1024],[0,724])
set(Y1,'Color','k')
hold on
Y2=ezplot('x^2-3.1396*x*y+2.2465*y^2 -25.7356*x -17.5923*y+7303.7=0',[0,1024],[0,724])
set(Y2,'Color','k')
hold on
Y3=ezplot('x^2-3.0242*x*y+2.1935*y^2-56.2710*x+ 20.2387*y+ 7303.6=0',[0,1024],[0,724])
set(Y3,'Color','k')
hold on
Y4=ezplot('x^2-2.3241*x*y+1.2907*y^2 -62.1475*x+ 27.1826*y+ 7303.7=0',[0,1024],[0,724])
set(Y4,'Color','k')
hold on
Y5=ezplot('x^2-1.9728*x*y+0.8818*y^2 -46.3128*x+10.0983*y+ 7303.9=0',[0,1024],[0,724])
set(Y5,'Color','k')
hold on
%}

%第五步：做切线
figure(4)
scatter(x1,y1,2)
hold on
plot([x_max(1) y_max(1)],[x_max(2) y_max(2)],'b')
hold on
plot([x_max(3) y_max(3)],[x_max(5) y_max(5)],'r')
hold on
plot([x_min(3) y_min(3)],[x_min(5) y_min(5)],'k')
hold on
plot([x_min(1) y_min(1)],[x_min(2) y_min(2)],'g')
hold on
%做过两切点中点的线
plot([(x_min(1)+x_max(1))/2                    (y_min(1)+y_max(1))/2],[(x_min(3)+x_max(3))/2
(y_min(3)+y_max(3))/2],'k')
hold on
plot([(x_min(3)+x_max(3))/2                    (y_min(3)+y_max(3))/2],[(x_min(5)+x_max(5))/2
(y_min(5)+y_max(5))/2],'k')
hold on
%做过两切点的线
plot([x_min(1) y_min(1)],[x_max(1) y_max(1)],'k')
hold on
plot([x_min(2) y_min(2)],[x_max(2) y_max(2)],'k')
```

```
hold on
plot([x_min(3) y_min(3)],[x_max(3) y_max(3)],'k')
hold on
plot([x_min(4) y_min(4)],[x_max(4) y_max(4)],'k')
hold on
plot([x_min(5) y_min(5)],[x_max(5) y_max(5)],'k')
hold on
%求直线的方程
[k12,b12]=solve('(y_min(1)+y_max(1))/2+((x_min(1)+x_max(1))/2)*k+b=0','(y_min(2)+y_max(2
))/2+((x_min(2)+x_max(2))/2)*k+b=0');
l12=@(x)k12*x+b12;
[k35,b35]=solve('(y_min(3)+y_max(3))/2+((x_min(3)+x_max(3))/2)*k+b=0','(y_min(5)+y_max(5
))/2+((x_min(5)+x_max(5))/2)*k+b=0');
l35=@(x)k35*x+b35;
[k1,b1]=solve('y_min(1)+x_min(1)*k+b=0','y_max(1)+x_max(1)*k+b=0');
l1=@(x)k1*x+b1;
[k2,b2]=solve('y_min(2)+x_min(2)*k+b=0','y_max(2)+x_max(2)*k+b=0');
l2=@(x)k2*x+b2;
[k3,b3]=solve('y_min(3)+x_min(3)*k+b=0','y_max(3)+x_max(3)*k+b=0');
l3=@(x)k3*x+b3;
[k4,b4]=solve('y_min(4)+x_min(4)*k+b=0','y_max(4)+x_max(4)*k+b=0');
l4=@(x)k4*x+b4;
[k5,b5]=solve('y_min(5)+x_min(5)*k+b=0','y_max(5)+x_max(5)*k+b=0');
l5=@(x)k5*x+b5;
%求两直线的交点
[x11,y11]=solve('y=k1*x+b1','y=k12*x+b12','x','y')
[x22,y22]=solve('y=k2*x+b2','y=k12*x+b12','x','y')
[x33,y33]=solve('y=k3*x+b3','y=k35*x+b35','x','y')
[x44,y44]=solve('y=k4*x+b4','y=k35*x+b35','x','y')
[x55,y55]=solve('y=k5*x+b5','y=k35*x+b35','x','y')
```