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）：A

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）：大连大学

参赛队员 (打印并签名) ：1.周吕文

2.苏小超

3.冯舒婷

指导教师或指导教师组负责人 (打印并签名)：刚家泰

日期：2008年 9 月 22 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）：

数码相机定位系统的数学模型

摘　要

数码相机定位在交通监管等方面有着广泛的应用，利用双目定位视觉测量系统可以迅速、准确地寻找搜寻物的特征点。但其准确度依赖于数码相机定位的准确度。因此，能否快速准确的对数码相机定位显得尤为重要。

对于问题一，本文首先确定了当圆面与主光轴不垂直时，所成像为非椭圆（下文称为伪椭圆）。然后利用简单的几何运算，对圆面与主光轴不垂直时的情况进行了细致的分析，最终确定了靶标上圆的圆心在相机像平面上所成的像应为该伪椭圆**最小外接圆**的圆心。

对于问题二，为了确定靶标上圆的圆心在像平面上的像坐标，本文将像平面根据分辨率分成了 1024×768 个格子来处理，运用**图像识别**的方法，建立了图形矩阵。在此基础上，取出了图像的边界，得到图像矩阵，设计了**最大距离矩阵算法**，计算出了靶标上各圆圆心对应的像坐标（单位：mm）依次为：$A(221.03,51.19)$, $B(247.35,49.47)$, $C(304.63,45.24)$， $D(210.85,-31.35)$ $E(289.68,-31.61)$

对于问题三，本文首先分析得出了像坐标的计算误差主要来源于像平面上离散的像素点。在图形矩阵的基础上，借鉴**元胞自动机**的思想，给出了**放大规则**，粗略的给出了各圆心像坐标误差范围的上界，并对各伪椭圆进行了 10 次放大，得到伪椭圆心坐标变化幅度很小的结论，即模型的稳定性较好。接着，本文对误差进行了更细致的考虑，提出了**像素细分**，通过计算四种极端情况最终得到了误差大约为 0.5 个像素。从而验证了模型具有高精度和强稳定性。

对于问题四，为了得到相机与靶标平面的空间位置关系，本文将复杂的空间几何运算简化为平面运算，逐次定出靶标平面上特征点的空间位置。然后本文以题目中靶标的像作为算例，进行了简单的编程运算，最终构建出确定两相机间相对位置的数学模型。

关键词：数码相机定位；最小外接圆；最大距离矩阵；放大规则 ；像素细分

一、问题重述

数码相机定位在交通监管等方面有广泛的应用，常用的定位方法是双目定位，即用两部相机定位，对物体上一个特征点，用两部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，用几何的方法可该特征点在固定一部相机的坐标系中的坐标，即确定特征点的位置。因此，精确确定两部相机的相对位置是关键，此过程称为系统标定。

标定的一种做法：在一平板上画若干点，同时两部相机照相，分别得到这些点在像平面上的像点，利用两组像点的几何关系可得到两相机的相对位置。但无论在物平面或像平面上本文都画出没有几何尺寸的“点”。实际做法是在物平面上画若干圆（称为靶标），圆心就是几何的点。而圆得像一般会变形，故须从靶标上的这些圆的像中把圆心的像精确地找到。

靶标设计：取边长为 100mm 的正方形，以四个顶点（A、C、D、E）为圆心 12mm 为半径作圆。以 AC 边上距 A 点 30mm 处 B 为圆心 12mm 为半径作圆（左图）。 用一位置固定的相机摄得其像（右图）。

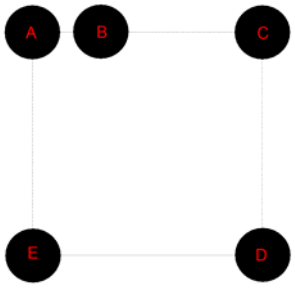


图 1

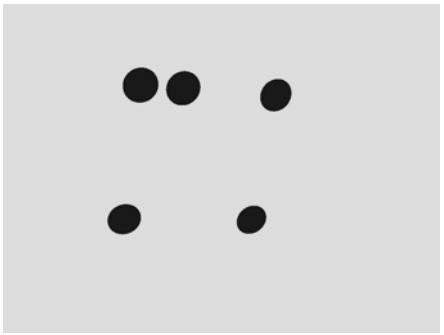


图 2

本文需要完成的题目：

（1）建立数学模型以确定靶标上圆的圆心在该相机像平面的像坐标，坐标系原点取在该相机的焦点，x-y 平面平行于像平面；

（2）由上图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（即焦点到像平面的距离）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，讨论方法的精度和稳定性；

（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

问题一

对于用于定位系统的数码相机，物体经过相机凸透镜所成的像，大多会因为拍摄角度而引起变形。当靶标上的圆与相机凸透镜的主光轴垂直时，所成的像是标准圆；但当圆与凸透镜的光轴不垂直时，所成的像则会因为不同的倾角而产生不同程度的变形。为方便叙述，本文作了如下定义：

- 靶标上的圆经过凸透镜成像后变形的圆称为**伪椭圆**（下文我们将分析，圆经过凸透镜后的图形并非为椭圆）；

- 靶标上圆的圆心在该像平面上所成的像点称为**伪椭圆的心**。

为了通过建立坐标系来确定伪椭圆的心坐标，寻找伪椭圆心的准确位置，设计一个有效的算法成为本文解决问题一的关键。**一个有效的数学模型和算法**应该满足以下条件：

- 有较高的执行效率，能够快速、准确地寻找伪椭圆的心的位置；并且易于计算机的实现；
- 对于一般情况和物殊情况都适用。当靶标上的圆与相机的凸透镜主光轴垂直时，成像为标准圆的特殊情况也是适用的。

问题二

问题一基本实现了确定伪椭圆心的位置的模型和算法，在此基础上，为了确定各靶标上圆心的像坐标，可以利用**图像识别**的方法，根据相机的分辨率对像平面进行分割，将像平面上的图形用矩阵形式表示出来。

同时，利用问题一中确定伪椭圆心位置的算法，可以在像素坐标系中计算各个伪椭圆的心的具体坐标。伪椭圆心的像素坐标系一旦确定，我们即可利用像素与长度的转换关系在像平面坐标系中确定各靶标上圆心在像平面的像坐标。

问题三

问题三要求设计一种方法来检验模型的精确性和稳定性，需要我们估计所建模型求得的伪椭圆的心像坐标的**误差范围**，只要该误差范围足够小，就可以说明模型的精度较好；对于模型稳定性的处理，可以考虑伪椭圆的心像坐标随着其它一些参数变化而变化的幅度。总之，选取一个直观的角度来设计一个好的方法确定误差范围是本问的关键。

问题四

问题四要求计算两部固定相机的相对位置，就要知道每部相机与靶标的空间位置关系。而每部相机与靶标的空间位置取决于相机的拍摄角度和拍摄距离，而靶标的空间位置与成像情况一一对应，因此，我们可以根据靶标的成像情况反推确定靶标的空间位置，这使得靶标空间位置与成像的相互关系的分析和计算成为本问的重点。

三、模型假设

- 假设相机凸透镜的尺度不计，即确定物的像时只考虑通过光心的光线。如果考虑相机的凸透镜的尺度，将使成像问题变得很复杂；
- 假设要求定标的两部相机的主光轴平行；

四、模型建立

4.1 问题一

为研究问题的方便，本文将数码相机成像看作简单的凸透镜成像，并假设相机的凸透镜的尺度不计，即确定物的像时只考虑通过光心的光线。由于拍摄角度等原因，一般情况下，数码相机所成的像都会出现不同程度的变形[1]。因此，本文在考虑相机所成像时，需要考虑拍摄角度和拍摄距离对相机所成像的影响。

圆面垂直于主光轴

当圆面与主光轴垂直时，如图 3 所示。由简单的几何光学知识及相似三角形性质可知，光线经过凸透镜所成的像仍然为圆形，不会发生变形。

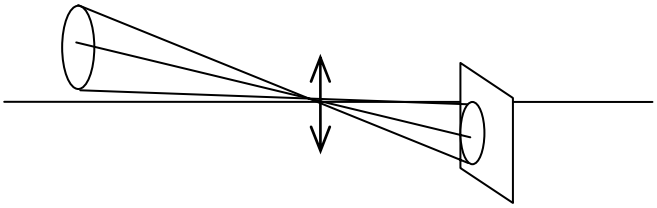


图 3 垂直主光轴的成像光路图

圆面不垂直于主光轴

照相机在实际照相进行双目定位的过程中，一般情况下，圆面与主光轴不能总是保持垂直，即圆与主光轴有一定的倾角。在这种情况下，相机所成像会出现不同程度的变形。

本文研究一般情况，即圆与主光轴并不垂直时，如图 4 所示。从侧面看，可将圆近似看作直径 AB， AB 的中点 C 为圆心。

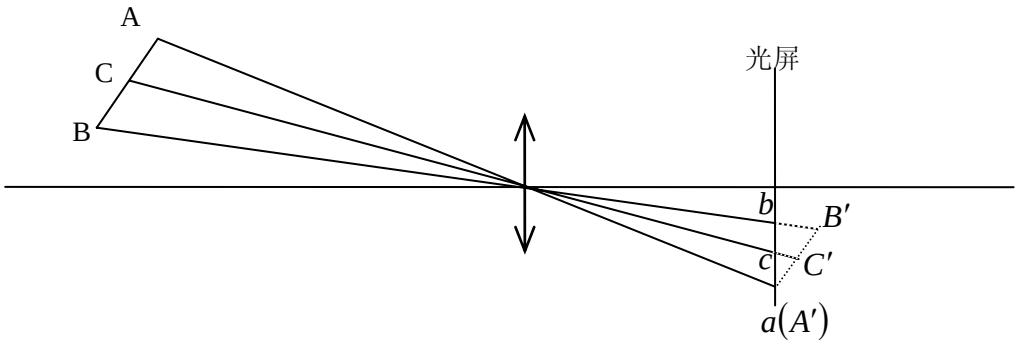


图 4 圆面与主光轴不垂直时的成像光路图

假设存在一个光屏 $A'B'$，使得：

$$A'B' \,//\, AB$$

由相似三角形的性质，可知：

$$A'C' = C'B'$$

由于这种情况下，圆面与主光轴不垂直，致使光屏并不与 AB 平行，这使得 c 并非为 ab 的中点，由简单的三角形相似知识可知：

$$bc < ca$$

c 相对于 ab 的中点位置上移了，因此，当圆与主光轴不垂直时，靶标上圆的圆心所成的像并非所成像的中心。由此可知，圆面经过相机的所成像并不是标准圆，而且由于圆心所成像位置的偏移，在成像的过程中，直径 AB 与其所成像 ab 的比例不同，即：

$$\frac{AC}{CB} \neq \frac{bc}{ca}$$

由此可知，圆面经过凸透镜成像后所成的像产生了变形，从几何角度来分析，该圆面经过凸透镜成像后变形所得的图形并非椭圆，本文称之为**伪椭圆**。

伪椭圆心的位置确定

为了确定伪椭圆心的位置，需要寻找圆面圆心的像点。对于一般情况，即圆面与主光轴不垂直时，过圆心且垂直于主光轴的平面，会与该圆面相交于该圆的一条直径，可知，该直径垂直于主光轴，如图 5 所示。由图 3 可知：垂直于主光轴的直径经过凸透镜后成像为伪椭圆的最长轴，且圆面的圆心（直径的中点）成像后成为最长轴的中点。

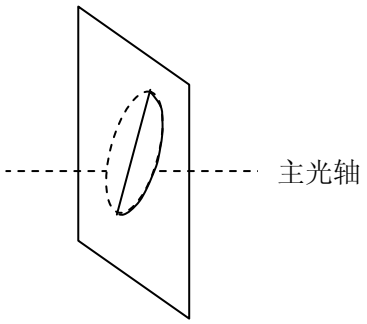


图 5 垂直于主光轴的平面与圆面相交于直径

由以上分析可知，圆面经过成像变形后，圆的大致形状如图 6 所示。与主光轴垂直的直径经过凸透镜成像后在像平面上形成了伪椭圆的最长轴，同时直径中点成像后成为最长轴的中点 o；在最短轴上，o 并非为其中点，而较中点向上产生了偏移。由此，本文可以很容易确定伪椭圆的心 o，**伪椭圆心即为伪椭圆最长轴的中点**。

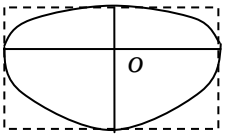


图 6　成像变形后的圆

对于 o 点坐标的确定，本文通过一个尽可能小的圆，如图 7 所示，将伪椭圆完全包含其中，最小外接圆的圆心即为本文需要确定的伪椭圆的心。从另外一个

角度来看，伪椭圆最长轴的中点为伪椭圆的心，相当于以最长轴为直径作了一个外接圆，**外接圆的圆心即为伪椭圆的心**。

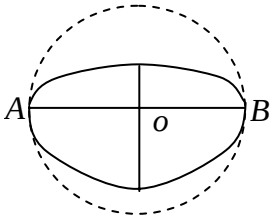


图 7　最小外接圆

我们求得最长轴的两端点为 $A(x_1,y_1),B(x_2,y_2)$，则伪椭圆心的坐标 $o(x,y)$ 为

$$x=\frac{x_1+x_2}{2},\quad y=\frac{y_1+y_2}{2}$$

4.2 问题二

对于相机而言，分辨率大小代表了像平面上所含像素点的多少，因此，在问题一的模型基本确定了计算靶标上圆的圆心在像平面上像坐标的方法的基础上，问题二只需利用给出的分辨率的值（1024×768），将像平面分割为 1024×768 个小格来处理，分割后的像平面，可以将像素作为坐标单位来确定伪椭圆心在像平面的位置，从而确定像平面上 5 个伪椭圆的心的像坐标。

图像识别

数码相机拍出来的照片上的图形并非连续的，而是由一个个像素点组成的。本文运用图像识别的思想，为方便问题的处理，利用直观的矩阵[2]来表示像平面中图形的像素点，如图 6 所示。其中，1 表求图形中伪椭圆的像素点，0 表示图形中非伪椭圆的像素点。

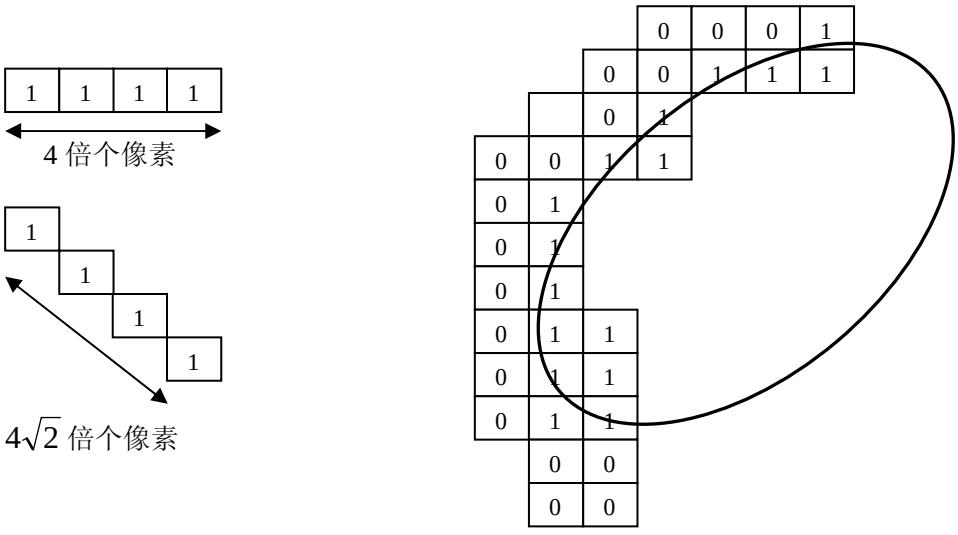


图 8 图形矩阵

边界矩阵

为了减小下文中最大距离矩阵的计算量，我们引入了边界矩阵。以得到图像的边界（算法见附录Boundary.m）。从矩阵变换到边界矩阵的示意图如图9

B

0	0	1	0	0
0	1	1	1	0
1	1	1	1	1
1	1	1	1	0
0	1	1	0	0

B 的边界矩阵

0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	0	0	1	0
0	1	1	0	0

图 9 边界矩阵示意图

用此算法，对我们要算的图像进行处理，以得到新的图像矩阵。题目中的图1 按照这种算法很容易得到其边界。

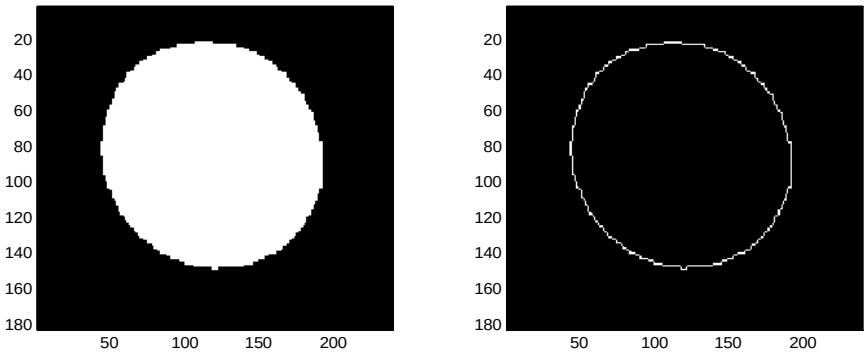


图 10　对图 1 进行边界矩阵算法处理前后

最大距离矩阵

按照问题一中模型的算法，为计算伪椭圆最小外接圆的圆心以及半径，本文引入了最大距离矩阵，最大距离矩阵就是把图形矩阵为 1 的值替换为它到整个伪椭圆上其它为 1 的点的最大距离值。利用这种思路，可以将问题一的模型引入该算法中，具体算法如下：

最大距离矩阵的算法

```
Fig 是图形矩阵
x 是图形矩阵中为元素为 1 的所在行
y 是图形矩阵中为元素为 1 的所在列
L 是 y 中元素的个数
R0=0
for i=1：L
    for j=1：L
        R=点(xi，yi)到(xj，yj)点的距离
        if R>R0
            R0=R
        end if
    end for
    Fig(yi，xi)=R
    R0=0
end for
```

即：对于图形边缘上的点，每一个点对应唯一坐标(x_i, y_i)，

$$r_i = \max\left\{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\right\} \quad j = 1,2\cdots, L$$

可得：

$$R = \max\{r_i\}$$

我们应用 *matlab* 软件进行计算，可得伪椭圆最长轴，从而确定相应的(x_i', y_i')，(x_j', y_j')，从而确定相应的圆心进而得到伪椭圆的圆心。

借助于Matlab编程很容易实现该算法，程序见附录 max_space_between.m。利用此最大矩阵算法，可以迅速、准确地寻找到最小外接圆的圆心，并且利用该程序运行效率较高，计算机总运算的时间不到0.3秒钟。

题目中的图1按照这种算法很容易求出其伪椭圆最小外接圆所对应的圆心。具体图形如图11所示

同理，问题二要求利用给出的靶标及其像来计算靶标上圆的圆心在像平面上的像坐标，我们同样利用该算法在分割后的像平面坐标系中确定伪椭圆的心在像素坐标系只的坐标值。

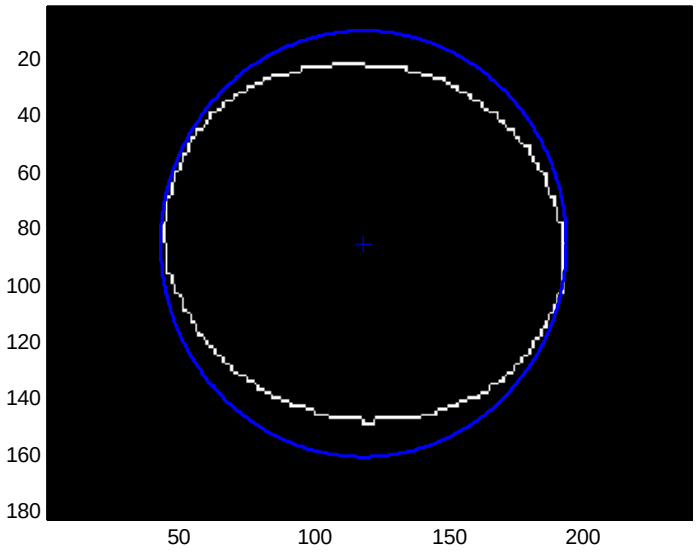


图 11　对图 1 进行最小外接圆圆心计算的结果

像坐标的确定

在像平面中，对于问题二所给的图形，本文利用最大距离矩阵算法，先将各个伪椭圆独立分析，即将各个伪椭圆置于独立的像素坐标系中来考虑，分别求出各自的心在像素坐标系中的像坐标（单位为像素）；然后再将五个伪椭圆综合起来，置于同一个像素坐标系中考虑，从而计算出各个伪椭圆的心在像素坐标系中的像坐标。

各伪椭圆的心在像素坐标系中如下图所示：

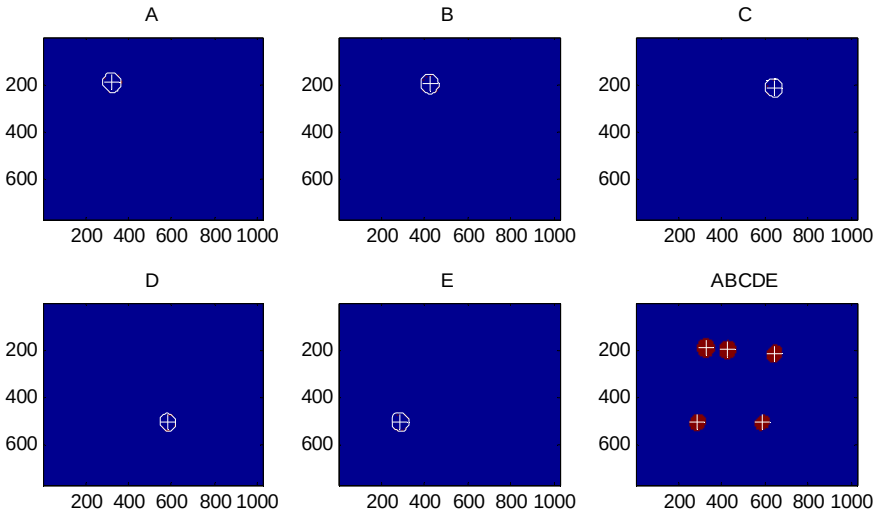


图 12 各个伪椭圆圆心坐标（像素）

图 12 中，如前所述，前 5 个像素坐标单独计算了各伪椭圆心的像坐标，最后将各单独考虑的伪椭圆综合置于同一个像素坐标系中，计算出了其对应的像素坐标。

同时， matlab 算法程序返回了各伪椭圆的心像坐标值，具体坐标值如下表所示：

表 1 伪椭圆的心坐标 单位：像素

伪椭圆	*A*	*B*	*C*	*D*	*E*
x	323．5	423．0	639．5	583．0	285．0
y	190．5	197．0	213．0	503．5	502．5

根据像素与长度的关系，我们可以将像素坐标系转化为长度坐标系来确定其在像平面中的坐标值，同时将像平面的中心作为坐标系的原点，将像素坐标系转化为距离坐标系。

经过坐标换算及单位换算：

$$x' = \left(x + 1024/2\right)/3.78$$

$$y' = -\left(y - 768/2\right)/3.78$$

可以得到以 mm 为单位，以光心为原点的各点的坐标。

表 2 伪椭圆的心坐标（保留小数点后两位） 单位：mm

伪椭圆	*A*	*B*	*C*	*D*	*E*
x	221.03	247.35	304.63	210.85	289.68
y	51.19	49.47	45.24	-31.35	-31.61

4.3 问题三

引起误差的原因

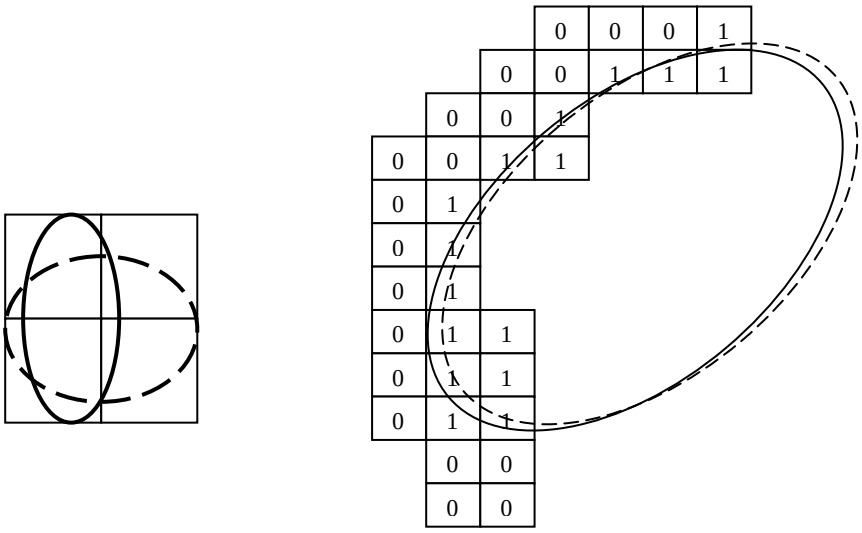
经分析，引起误差的原因主要有以下两个原因：

- 本文在处理图片时，将对应矩阵属于伪椭圆的位置赋为 1，否则赋为 0，但在伪椭圆边界处的一些点并不能很清楚的判定其该赋值为 1 或 0，我们仅人为的设定了一个阈值，来区分哪些位置该赋为 1，哪些位置该赋为 0。因此这也将造成一定的误差，图 13 是处理前后图像的对比。

原图 0－1化后的

图 13 处理前后图像的对比

- 由于数码相机拍出来的照片都是由像素点组成的，如图 14 所示，只要被图形覆盖的像素点均会反映出图形，由于每个像素点都有一定的长度和宽度，故不同图形很可能会对应相同的像素点，这使得数码相机拍出的像并不能完全确定，两个不重合的伪椭圆却可以对应着同一张相片。



(a)　　(b)

图 14 图形与像素关系图

因此，由于**离散的像素**原因，问题二中我们利用分割像平面所求得的伪椭圆的心与实际圆心所成的像点可能存在着一定程度的误差。

利用本文所建模型的思路，最长轴的中点即为伪椭圆的心，同样会由于像素的原因而使得计算所得的伪椭圆的心存在着一定的偏差。故从像素角度来寻求一种检验模型准确性和稳定性的方法，是比较合理的。

为了检验模型的精度和稳定性，我们需要对所建模型的误差范围进行确定。

误差的分析

对模型误差的估计，利用题目中的已知条件，*A*、*B*、*C* 三个圆面的圆心在同一条直线上。利用凸透镜成像的原理，一条线段经过凸透镜成像后仍为线段，因此，A、B、C 三个圆的圆心所成的像也应在一条直线上。

但是由于像素离散所造成的误差，可能使我们计算出的三点的坐标并不严格的在同一条直线上，作 A、B、C 三个伪椭圆心的两两连线图，如图 15，显然 A、B、C 并不严格在同一直线上，这证明了我们猜测的正确性。

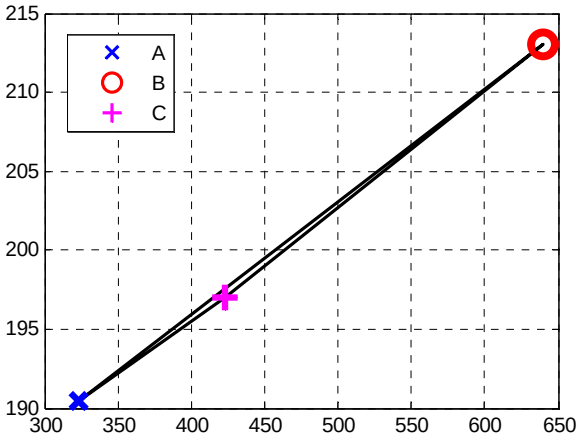


图 15　A、B、C 三点坐标连线图

为估计误差，我们对三个点进行直线拟合，三个点到拟合直线的距离（单位：像素）分别为：0.0015、0.0022、0.0007。

三个点到拟合直线的距离都远远小于一个像素，某种程度上说明了我们**模型的误差相对而言是比较小的，模型的精度较高。**

下面本文将设计一种算法来估计伪椭圆的心像坐标的误差：

误差范围的确定

为了讨论模型的精度和稳定性，我们需要对模型的误差范围进行确定。要考虑系统的精确度，就是要考虑不确定性的大小，我们通过对伪椭圆边界的考虑来确定系统的不确定性大小。

从图 14 中我们可以发现，不同的图形可对应着同一种像素分布，要考虑伪椭圆的心坐标的精度，就要考虑伪椭圆心的可移动范围，对于伪椭圆心的可移动范围实际上就是其外接圆圆心的可移动范围。因为同一种像素分布在问题二的算法中对应着同一个外接圆，因此我们需要对伪椭圆边界做很小的放大，使得放大后的边界的范围是原伪椭圆可能移动的范围，从而确定误差范围的上界。所以，最终采用通过将原来的伪椭圆向外扩大的方法，来确定实际计算所得的圆心的波动范围。

我们借鉴了**元胞自动机**思想，给出以下具体的**放大规则**：

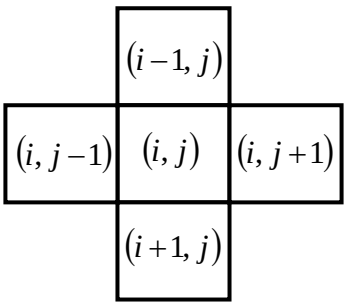


图 16 伪椭圆放大规则图

如图 16，按照分辨率分割的像平面图形矩阵，对于图形矩阵中的(i,j)点，如果它本身的值为 0，且周围四个元素中若有一个或一个以上值为 1 时，它将变为 1,即

$$(i,j)=\begin{cases}0 & if \quad (i-1,j)+(i+1,j)+(i,j-1)+(i,j+1)+(i,j)=0\\ 1 & else\end{cases}$$

按照此种放大规则，则可知图 13 将转变为图 16(a)。图 16(a)中的虚线所包围的范围即为原伪椭圆的可移动范围，虚线中所包围的伪椭圆即为实际伪椭圆可能的位置之一。

图 17(b)中的圆即为(a)中伪椭圆所对应的最小外接圆，利用原伪椭圆的最小外接圆在扩大后伪椭圆的最小外接圆内的可移动情况，我们即可确定实际伪椭圆的心的范围。

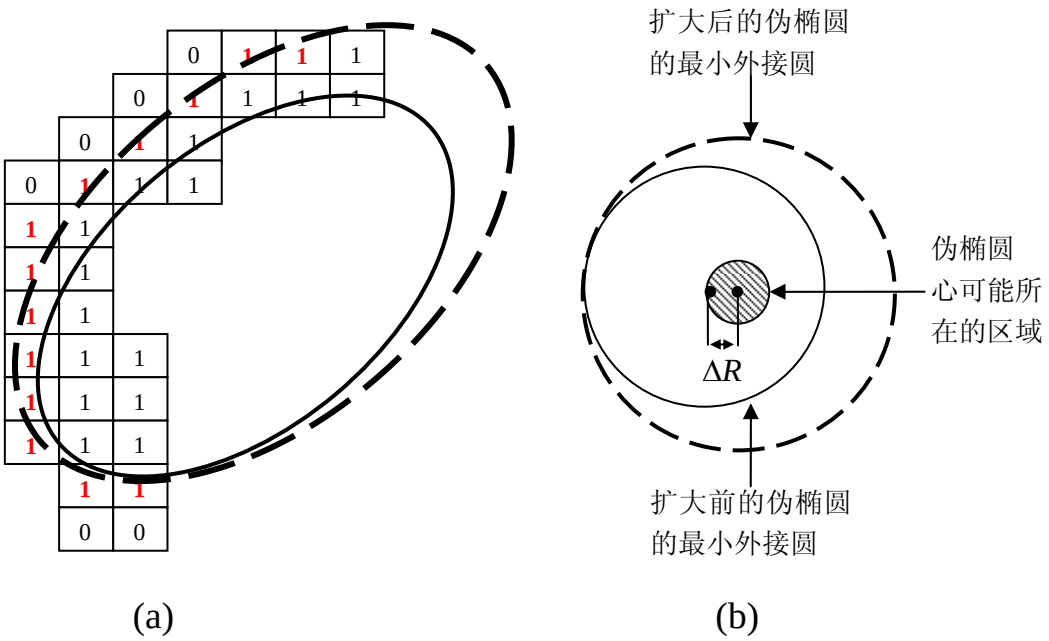


图 17 向外扩大一个像素的伪椭圆与最小外接圆

实际的伪椭圆在虚线内，但位置不确定，对应的伪椭圆的最小外接圆也在虚线伪椭圆的外接圆中。因此只要知道虚线伪椭圆的外接圆半径与伪椭圆的外接圆半径之差，就可能知道伪椭圆心的可移动范围。

$$\Delta R = R - r$$

其中r, R，分别是伪椭圆扩大前后的最长半轴。

精度

由第三问的算法，本文分别求出扩大前后的伪椭圆的外接圆半径，两次所得半径之差，即为伪椭圆心的所在区域的半径。

由此，我们可以求得五个伪椭圆心的可移动范围半径，具体伪椭圆心的可移动范围半径见表 3。

表 3　五个伪椭圆心的可移动范围半径　　单位：像素

伪椭圆	A	B	C	D	E
ΔR	0.8646	0.8500	0.8597	0.7951	0.9327

从上表我们可以看出，5 个伪椭圆心的可移动范围半径虽然小于 1 个像素，但都较接近于 1 像素，这很可能是因我们放大规则引起了误差的扩大，因此，我们认为实际的误差半径小于ΔR，后面我们将对误差作更细致的考虑。

稳定性

当伪椭圆的边界范围发生较小的变化，伪椭圆心的坐标不发生较大程度的变化时，我们认为模型是稳定的，反之，模型不稳定。

因此，对于稳定性的处理，我们只要利用第二问求伪椭圆心的像坐标的算法，对放大前后伪椭圆的心像坐标作比较、分析，即可知道像坐标是否变化显著，从而判断该模型的稳定性。

因此，本文在第二问的基础上，对 5 个伪椭圆做了 10 次放大，来判断其心的像坐标是否发生较大的变化。5 个伪椭圆的心像坐标值及相对变化如下表所示：

表 4 五个伪椭圆扩大十次前后的心坐标值及变化

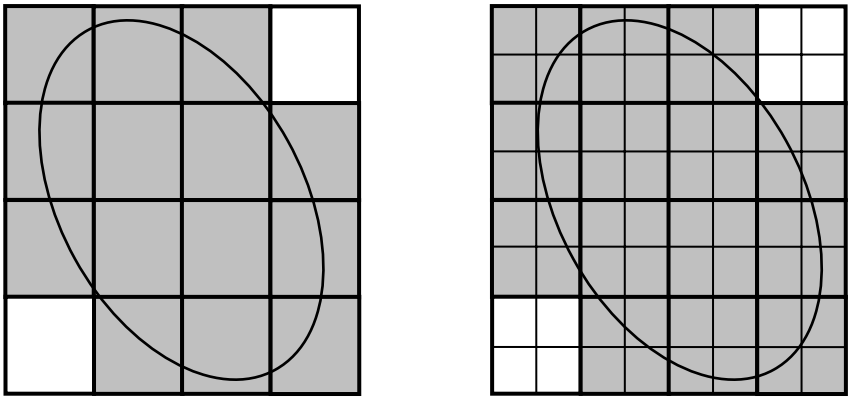
近似椭圆	放大前坐标		放大后坐标		差值	
	x	y	x	y	Δx	Δy
A	323.5	190.5	322.0	187.5	-1.5	-3.0
B	423.0	197.0	419.5	190.5	-3.5	-6.5
C	639.5	213.0	639.5	213.0	0.0	0.0
D	583.0	503.5	582.5	502.5	-0.5	-1.0
E	285.0	502.5	284.5	501.0	-0.5	-1.5

从上表可知，经过 10 次放大后伪椭圆的心像坐标发生变化的幅度并不是很大，由此，我们可以确定伪椭圆的心坐标不会随着一些参数的变化而大幅度地变化，故可说明**伪椭圆的心坐标的算法有一定的稳定性**。

更细致误差的考虑

上文中，我们提出的放大规则对误差的考虑过于粗糙，下面我们提出一种更细致，更精确的算法来确定误差。

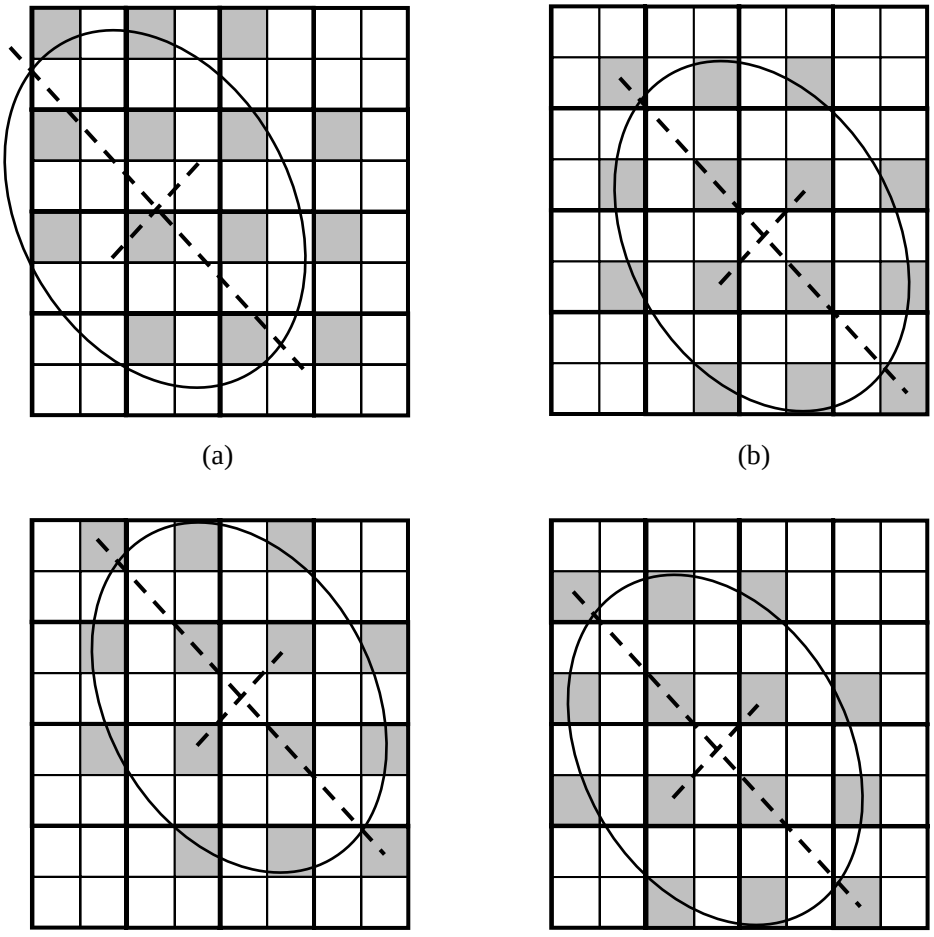
为了更得到更细致的像素图像，本文提出了对像素的细分思想，把一个像素细分为L^2个像素（本文称为L次细分），如图18(b)图是对(a)的二次细分。当然，在计算机里，像素的细分是反应在矩阵的扩充上的（表示图(a)的4×4的矩阵扩充为16×16的矩阵），相关算法见`fractionize_Pixels.m`。



(a) (b)

图 18 像素的二次细分

对于细分后的像素，我们在每一个原像素细分成的L^2个像素中，取一个有效相素点，便可以很容易得到四种极端的情况（二次细分的四种情况见图19），相关算法见Eccentricity.m。这四种极端情况的伪椭圆心在各个方向上偏离最大，但对应的同一个原相素图（图19中的四个图对应的原像素图像都是图18中的(a)图）。



(a) (b)

(c) (d)

图 19 像素的二次细分后的四种偏离

我们通过计算细分后的四种极端情况的伪椭圆心坐标，便可较精确的估算出误差。当然细分的次数 L 越大，估算的越精确，但同时也会增大计算机的运算量，在此本文仅取 $L=5$。表 5 是细分后各伪椭圆四种极端情况的心坐标。

表 5 细分后四种极端情况的伪椭圆心坐标

近似椭圆	a 型		b 型		c 型		d 型	
坐标	x_a	y_a	x_b	y_b	x_c	y_c	x_d	y_d
A	322.7	189.7	323.5	189.7	322.7	190.5	323.5	190.5
B	422.2	196.2	423.0	196.2	422.2	197.0	423.0	197.0
C	638.7	212.2	639.5	212.2	638.7	213.0	639.5	213.0
D	284.2	501.7	285.0	501.7	284.2	502.5	285.0	502.5
E	582.2	502.7	583.0	502.7	582.2	503.5	583.0	503.5

在求得四种极端情况下的伪椭圆心 $(x_a,y_a),(x_b,y_b),(x_c,y_c),(x_d,y_d)$ 后，容易得到伪椭圆心的可移动半径为：

$$\Delta R=\frac{1}{2}\sqrt{(x_a-x_d)^2+(y_a-y_d)^2}\ \text{或}\ \frac{1}{2}\sqrt{(x_b-x_c)^2+(y_b-y_c)^2}$$

根据以上公式，可求得五个伪椭圆心的可移动范围半径，结果见表 6。

表 6 更细致考虑后的五个伪椭圆心的可移动范围半径　　单位：像素

伪椭圆	A	B	C	D	E
ΔR	0.5657	0.5657	0.5657	0.5657	0.5657

显然，表 6 得到的结果要比本文先前放大规则得到的可移动范围半径（见表 3）小的多，这也证实了先前我们的猜想：放大规则引起了误差的扩大。最终，我们认为误差半径约为 0.5 个像素。

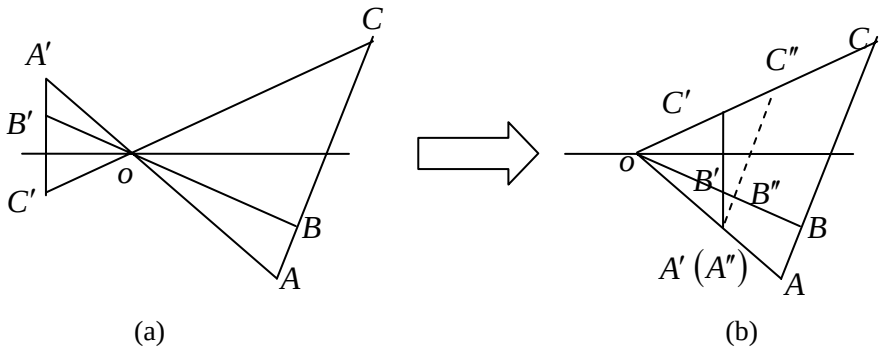
4.4 问题四

通过靶标来确定两部固定相机的相对位置，实际上是双目定位的一个逆过程，因此，我们如果可以求得靶标上的圆心与坐标原点的距离，以及靶平面与坐标平面的倾角关系，就可以求得两部固定相机间的相对位置。

我们为了回避复杂的空间几何关系和运算，将其转化为平面几何关系来分析凸透镜成像过程，如下图所示。A、B、C 三个圆面的圆心在同一直线上，所成像 A'、B'、C' 也应在同一直线上，我们将图 20 中(a)图转化为(b)图。显然可得：

$$\frac{AB}{BC}=\frac{A''B''}{B''C''}=\frac{3}{7}$$

同时，根据问题二对各圆心像坐标的计算，我们可以计算出 $A'B'/B'C'$。



(a) (b)

图 20 ABC 圆心成像光路图及简化图

以 AoC 所在平面建立直角坐标系，以主光轴在该平面的投影为 x' 轴，通过模型二， A',B',C' 三点的坐标均可以得到，设 A',B',C' 坐标分别 $(x_{A'},y_{A'})$，$(x_{B'},y_{B'})$，$(x_{C'},y_{C'})$。

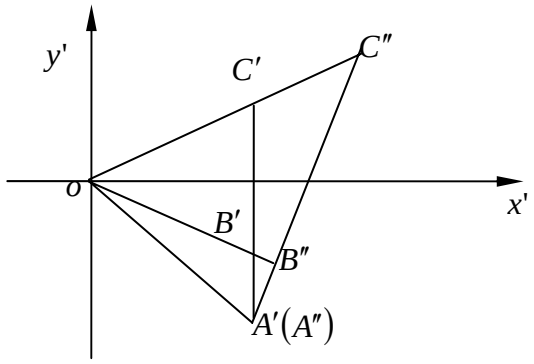


图 21 AoC 平面坐标系

oA',oB',oC' 的方程均可得到，分别如下：

$$y=\frac{y_{A'}}{x_{A'}}x,\quad y=\frac{y_{B'}}{x_{B'}}x,\quad y=\frac{y_{C'}}{x_{C'}}x$$

设 $A''C''$ 的方程为：

$$y-y_{A'}=k(x-x_{A'})$$

故 $A''B''$ 的长度 $A''B''(k)$ 及 $B''C''$ 的长度 $B''C''(k)$ 均可用 k 来表示，而

$$\frac{A''B''(k)}{B''C''(k)}=\frac{3}{7}$$

由此可解得 k，进而可得到各点的坐标：$B''(1553.0,-108.6)$，$C''(1481.0,103.53)$（相关程序见附录：Coordinate_of_C_D.m）。利用 $ox'y'$ 中的坐标，我们可以得到 oA'',oB'',oC'' 的长度，进而很容易求得各点对应的三维坐标（以焦点为原点，x-y 面平行于像平面）。

如图所示，黄色平面为像平面，红线为 $A''B''C''$ 的连线。

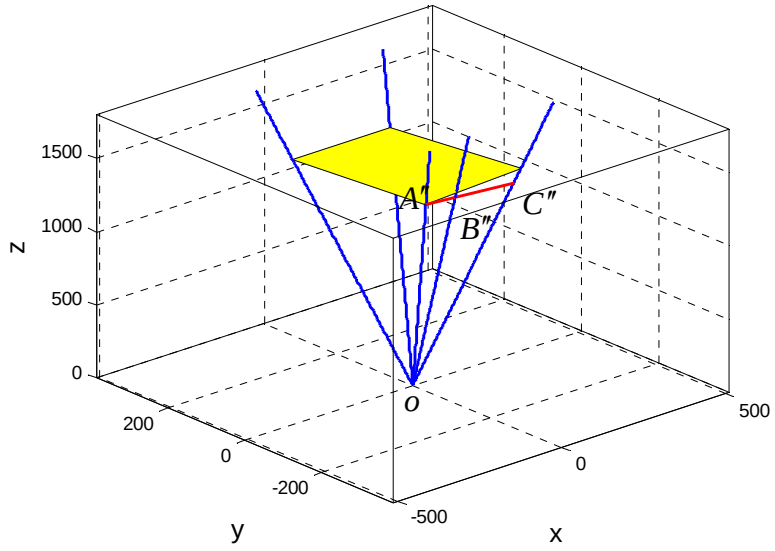


图 22　A''、B''、C'' 三维空间图

由于靶标是一个正方形，各边应相等，因此我们可以在三维坐标系中得到该正方形其它边的连线，如图22的红色连线，红色平面表示靶标平面。利用连线围成的近似平面（误差导致了连线不严格在同一个平面上）与靶标平面平行的特征，根据各边的比例关系，可以得到靶标上圆面各圆心在三维坐标系中的所在位置（相关程序见附录：**main_4.m**）。

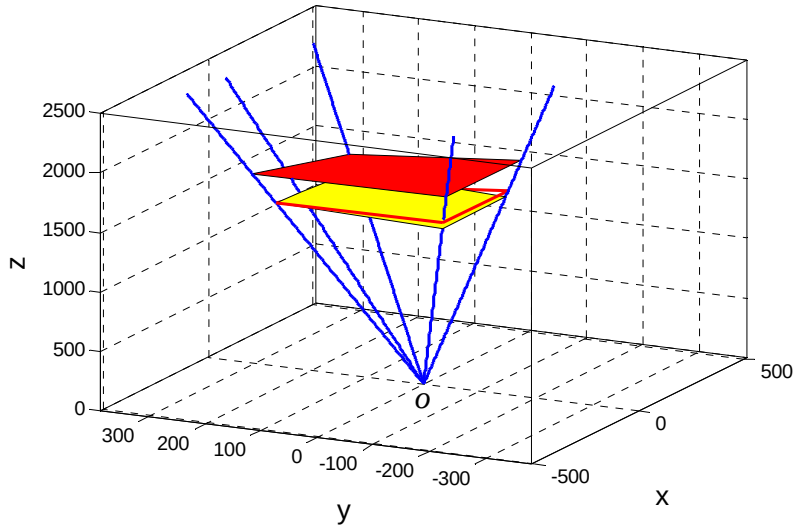


图 23　靶标平面三维空间图

如下图所示，根据靶标上各圆圆心在三维坐标系中的坐标，我们可以得到靶标上圆的圆心 S 与原点 o 的距离 l 以及 oS 与 z 轴的夹角 ϕ。

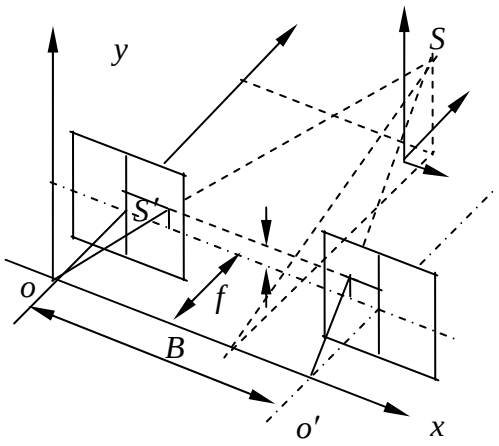


图 24 成像原理三维图

根据问题二求解所得 S 点的像 S' 的坐标 $\left(x_0',y_0',z_0'\right)$，以及经过上述分析所计算出的 oS 之间的距离 l_l 以及 oS 与 z 轴之间的夹角 φ_1，同理求得的 $o'S$ 之间的距离 l_2 以及 $o'S$ 与 z 轴之间的夹角 φ_2，我们可以进一步求解确定靶标圆心的位置：

设 S 点的坐标为 (x_0,y_0,z_0)，可知：

$$\begin{cases} \dfrac{\sqrt{x_0'^2+y_0'^2+z_0'^2}}{l_1}=\dfrac{y_0'}{y_0} \\ z_0=\sin\varphi_1 l_1 \\ \\ x_0=\sqrt{l_1^2-y_0^2-z_0^2} \end{cases}$$

从而可以确定出 S 点坐标。

在三角形 oBS 中

$$oB=\sqrt{l_1^2-y_0^2-z_0^2}$$

代入值可得

$$oB=\sqrt{l_1^2-\frac{x_0'^2+y_0'^2+z_0'^2}{l_1^2y_0'^2}-\sin^2\varphi_1 l_1^2}$$

为确定两相机之间的距离，我们只需确定另一台相机的坐标。由于两台相机的主光轴是平行的，故另一台相机也在坐标轴上。如图 $BS\perp x$ 轴，且 $oB=|x_0|$，在三角形 $o'BS$ 中，则可计算出 $o'B$ 的长度

$$o'B=\sqrt{l_2^2-y_0^2-z_0^2}$$

则两相机之间的距离

$$L=oB+o'B$$

从而确定两相机之间的距离 L。

五、模型评价

5.1 模型的优点

- 在对伪椭圆的心位置进行确定时，从凸透镜成像的本质出发，使模型简单、易懂；
- 对靶标上各圆的圆心所成像的像坐标进行计算时，运用了图像识别的思想，将像平面转化为离散的矩阵形式，从而有利于像坐标的读取。同时，坐标计算所运用的算法程序执行率高，运行时间短；
- 对模型的精度和稳定性进行讨论时，通过拟定放大规则，确定了误差的上界，从而保守地估计了实际误差值，更加能够检验模型的精度和稳定性。

5.2 模型的缺点

在解决第四问的时，为了避免太复的几何运算，本文利用计算机进行了相关的运算。如果要求计算机的运算精度很高时，将花费我们大量的时间和精力。因此，第四问的结果精度相对不是很高，这可能也是导致了连线不严格在同一个平面上的原因之一。

六、参考文献

[1] 李小彤，几何光学·像差·光学设计，计量学报， Vol.4，Nov，2003

[2] Luke Winstrom, Sam Coskey, Mark Blunk, Shelling Tumors with Caution and Wiggles, The UMAP Journal 24(3)(2003)367-380

[3]张之江，双摄像机靶标成像视觉坐标测量方法研究，测试技术学报，Vol.4，No.18,2004

附录

附录一：问题二

Main_1.m

```
%将图1保存为fig01.bmp放在同一个目录下
fig=imread('fig01.bmp'); %读入图像
fig=fig(:,:,1);
fig(fig<=70)=0;
fig(fig>70)=1;
fig=Boundary(fig)
Fig(:,:,1)=fig;
Fig(:,:,2)=fig;
Fig(:,:,3)=fig;
image(Fig*255)%显示图形
[p,r,cx,cy]=max_space_between(fig);
hold on
plot(cx,cy,'+b','markersize',8)
theta=0:0.01:2*pi;
X=cx+r*cos(theta);
Y=cy+r*sin(theta);
plot(X,Y,'b','linewidth',2);
```

Boundary.m

```
function boundary=Boundary(M)
[I,J]=size(M);
i=2:I-1;j=2:J-1;
SUM=M(i-1,j)+M(i+1,j)+M(i,j-1)+M(i,j+1);%Sum of four neighbours
m=M(i,j);
m(SUM==4)=0; % if all of (i,j)'s four neighbours==1 then m(i,j)=0
M(i,j)=m;
boundary=M;
```

max_space_between.m

```
function [p,r,cx,cy]=max_space_between(fig)
p=zeros(size(fig));%相对最大位置矩阵
[y,x]=find(fig==1);%找出图形的像素点
R=[];%图形内各点的最大相对位置
r0=0;
for i=1:length(x)
    for j=1:length(x)
        r=((x(i)-x(j))^2+(y(i)-y(j))^2)^0.5;
        if r>r0 % 找出离(xi,yi)最远的点
            r0=r;
```

```
        end
    end
    R=[R;r0];
    r0=0;
end

for i=1:length(x)
p(y(i),x(i))=R(i);%将离点(xi,yi)的最大距离放在矩阵对应的置
end

%找出p矩阵中图形像素点相距最大的两点
[y,x]=find(p==max(max(p)));
Rs=(x-x(1)).^2+(y-y(1)).^2;

%取一组（这里是为了防止有正圆的情况）
i=find(Rs==max(Rs));
x=x([1,i]);
y=y([1,i]);

r=sqrt((x(1)-x(2))^2+(y(1)-y(2))^2)/2%最小外接圆半径
cx=mean(x)%圆心位置的横坐标
cy=mean(y)%圆心位置的纵坐标
```

Main_2.m

```
%将图2保存为fig02.bmp放在同一个目录下
fig=imread('fig02.bmp'); %读入图像
fig=fig(:,:,1);
fig(fig<50)=1;
fig(fig>=50)=0;

%将每个伪椭圆分离
fig1=zeros(size(fig));
fig2=fig1;fig3=fig1;fig4=fig1;fig5=fig1;
fig1(1:300,200:370)=fig(1:300,200:370);
fig2(1:300,370:500)=fig(1:300,370:500);
fig3(1:300,550:700)=fig(1:300,550:700);
fig4(400:600,200:400)=fig(400:600,200:400);
fig5(400:600,500:650)=fig(400:600,500:650);
%取边界
fig1=Boundary(fig1);
fig2=Boundary(fig2);
fig3=Boundary(fig3);
fig4=Boundary(fig4);
```

```
fig5=Boundary(fig5);
%第一个伪椭圆的处理
subplot(2,3,1)
image(fig1*255)
[p1,r1,cx1,cy1]=max_space_between(fig1);
hold on
plot(cx1,cy1,'+w','markersize',8)
theta=0:0.01:2*pi;
X1=cx1+r1*cos(theta);
Y1=cy1+r1*sin(theta);
plot(X1,Y1,'w','linewidth',1);
title('1')

%第二个伪椭圆的处理
subplot(2,3,2)
image(fig2*255)
[p2,r2,cx2,cy2]=max_space_between(fig2);
hold on
plot(cx2,cy2,'+w','markersize',8)
X2=cx2+r2*cos(theta);
Y2=cy2+r2*sin(theta);
plot(X2,Y2,'w','linewidth',1);
title('2')

%第三个伪椭圆的处理
subplot(2,3,3)
image(fig3*255)
[p3,r3,cx3,cy3]=max_space_between(fig3);
hold on
plot(cx3,cy3,'+w','markersize',8)
theta=0:0.01:2*pi;
X3=cx3+r3*cos(theta);
Y3=cy3+r3*sin(theta);
plot(X3,Y3,'w','linewidth',1);
title('3')

%第四个伪椭圆的处理
subplot(2,3,4)
image(fig4*255)
[p4,r4,cx4,cy4]=max_space_between(fig4);
hold on
plot(cx4,cy4,'+w','markersize',8)
theta=0:0.01:2*pi;
X4=cx4+r4*cos(theta);
```

```
Y4=cy4+r4*sin(theta);
plot(X4,Y4,'w','linewidth',1);
title('4')

%第五个伪椭圆的处理
subplot(2,3,5)
image(fig5*255)
[p5,r5,cx5,cy5]=max_space_between(fig5);
hold on
plot(cx5,cy5,'+w','markersize',8)
theta=0:0.01:2*pi;
X5=cx5+r5*cos(theta);
Y5=cy5+r5*sin(theta);
plot(X5,Y5,'w','linewidth',1);
title('5')

%整体显示
subplot(2,3,6)
image(fig*255);
hold on
plot(cx1,cy1,'+w','markersize',8)
plot(cx2,cy2,'+w','markersize',8)
plot(cx3,cy3,'+w','markersize',8)
plot(cx4,cy4,'+w','markersize',8)
plot(cx5,cy5,'+w','markersize',8)
title('1-5')
```

附录二：问题三

error_analyse.m

```
R=[];X=[];Y=[];
for k=1:2
    fig=imread('fig02.bmp'); %读入图像
    fig=fig(:,:,1);
    fig(fig<50)=1;
    fig(fig>=50)=0;

    if k==2%进行放大后再求解
       %================放大规则==============
       %for k=1:10%扩大10次
       [I,J]=size(fig);
       i=2:I-1;j=2:J-1;
       SUM=fig(i-1,j)+fig(i+1,j)+ fig(i,j-1)+fig(i,j+1);%求周围四个位置和
       FIG=fig(i,j);
       FIG(FIG==0&SUM>0)=1;
       fig(i,j)=FIG;
       %end
    end

   %将每个伪椭圆分离
   fig1=zeros(size(fig));
   fig2=fig1;fig3=fig1;fig4=fig1;fig5=fig1;
   fig1(1:300,200:370)=fig(1:300,200:370);
   fig2(1:300,370:500)=fig(1:300,370:500);
   fig3(1:300,550:700)=fig(1:300,550:700);
   fig4(400:600,200:400)=fig(400:600,200:400);
   fig5(400:600,500:650)=fig(400:600,500:650);

   fig1=Boundary(fig1);
   fig2=Boundary(fig2);
   fig3=Boundary(fig3);
   fig4=Boundary(fig4);
   fig5=Boundary(fig5);

   [P1,R1,Cx1,Cy1]=max_space_between(fig1);
   [P2,R2,Cx2,Cy2]=max_space_between(fig2);
   [P3,R3,Cx3,Cy3]=max_space_between(fig3);
   [P4,R4,Cx4,Cy4]=max_space_between(fig4);
   [P5,R5,Cx5,Cy5]=max_space_between(fig5);
   R=[R [R1;R2;R3;R4;R5]];
   X=[X [Cx1;Cx2;Cx3;Cx4;Cx5]];
```

```
    Y=[Y [Cy1;Cy2;Cy3;Cy4;Cy5]];
end
dR=R(:,2)-R(:,1)
dX=X(:,2)-X(:,1)
dY=Y(:,2)-Y(:,1)
```

fractionize_Pixels.m

```
function M=fractionize_Pixels(m,n)
% 相素细分函数
% m原矩阵
% n细分的倍数

% a=[ 1     0
%     3     2]

% A=fractionize_Pixels(a,2)

% A=[ 1     1     0     0
%     1     1     0     0
%     3     3     2     2
%     3     3     2     2];

[x,y]=size(m);

%纵向细分
M1=zeros(n*x,y);
for i=1:n
    M1(i:n:(end-n+i),:)=m;
end

%横向细分
M2=zeros(n*x,n*y);
for i=1:n
    M2(:,i:n:(end-n+i))=M1;
end
M=M2;
```

Eccentricity.m

```
function MM=Eccentricity(m,n,L)
% L=1,2,3,4
% L==1往左上偏，L＝＝2往右上偏，
% L＝＝3往左下偏，L＝＝4往右下偏
M=fractionize_Pixels(m,n);
```

```
MM=zeros(size(M));
if L==1
    MM(1:n:end,1:n:end)=M(1:n:end,1:n:end);
elseif L==2
    MM(1:n:end,n:n:end)=M(1:n:end,n:n:end);
elseif L==3
    MM(n:n:end,1:n:end)=M(n:n:end,1:n:end);
elseif L==4
    MM(n:n:end,n:n:end)=M(n:n:end,n:n:end);
end
```

More_error_analyse.m

```
R=[];X=[];Y=[];
fig=imread('fig02.bmp'); %读入图像
fig=fig(:,:,1);
fig(fig<50)=1;
fig(fig>=50)=0;
%将每个近似椭圆分离
fig1=zeros(size(fig));
fig2=fig1;fig3=fig1;fig4=fig1;fig5=fig1;
fig1(1:300,200:370)=fig(1:300,200:370);
fig2(1:300,370:500)=fig(1:300,370:500);
fig3(1:300,550:700)=fig(1:300,550:700);
fig4(400:600,200:400)=fig(400:600,200:400);
fig5(400:600,500:650)=fig(400:600,500:650);

fig1=Boundary(fig1);
fig2=Boundary(fig2);
fig3=Boundary(fig3);
fig4=Boundary(fig4);
fig5=Boundary(fig5);

% 只需把fig1依次改为fig2,fig3,fig4,fig5,
% 就可得各个点的四个极端坐标

Fig11=Eccentricity(fig1,5,1);
[P11,R11,Cx11,Cy11]=max_space_between(Fig11);
clear P11 R11

Fig12=Eccentricity(fig1,5,2);
[P12,R12,Cx12,Cy12]=max_space_between(Fig12);
clear P12 R12
```

```
Fig13=Eccentricity(fig1,5,3);
[P13,R13,Cx13,Cy13]=max_space_between(Fig13);
clear P13 R13

Fig14=Eccentricity(fig1,5,4);
[P14,R14,Cx14,Cy14]=max_space_between(Fig14);
clear P14 R14

[Cx11 Cy11;Cx12 Cy12;Cx13 Cy13;Cx14 Cy14]/5
```

附录三：问题四

Coordinate_of_C_D.m

```
%求B"和C"的坐标
X=[];Y=[];
xa1=1586.5;
ya1=-205.92;

xb1=1586.5;
yb1=110.9;

xc1=1586.5;
yc1=110.9;

ab_bc=0.4593;%ab与bc的比值

xi=0:1:5000;
yi=110.9*xi/1586.5;

for i=1:length(xi)
    [x,y]=point_of_intersection([xi(i),xa1],[yi(i),ya1],[0,1586.5],[0,-110.9]);
    X=[X,x];
    Y=[Y,y];
end

r_ab=((xa1-X).^2+(ya1-Y).^2).^0.5;
r_bc=((xi-X).^2+(yi-Y).^2).^0.5;

r=r_ab./r_bc;
R=abs(r-ab_bc);

xc2=find(R==min(R))
yc2=110.9*xc2/1586.5

xb2=X(xc2)
yb2=Y(xc2)
```

point_of_intersection.m

```
function [x,y]=point_of_intersection(x1,y1,x2,y2)
%求过（x1(1),y1(1)）和（x1(2),y1(2)）的直线与
%过（x2(1),y2(1)）和（x2(2),y2(2)）的直线的交点

k1=diff(y1)/diff(x1);
k2=diff(y2)/diff(x2);
```

```
x=((k1*x1(1)-y1(1))-(k2*x2(1)-y2(1)))/(k1-k2);
y=k1*x-k1*x1(1)+y1(1);
```

main_4.m

```
x=[323.5 423.0 639.5 285.0 583.0];
y=[190.5 197.0 213.0 502.5 503.5];
%＝＝＝＝＝＝＝＝坐标变换＝＝＝＝＝＝＝＝
x=x-1024/2;
y=-y+768/2;
z=1577*ones(5,1);
X=[x(1:3),x(5),x(4)];
Y=[y(1:3),y(5),y(4)];

fill3(X,Y,z,'y')
box on
grid on
axis([-1024/2,1024/2,-768/2,768/2,0,2500])
xlabel('x','fontsize',13);
ylabel('y','fontsize',13);
zlabel('z','fontsize',13);
hold on

for i=1:5
plot3([0:x(i)/1000:1.6*x(i)],[0:y(i)/1000:1.6*y(i)],[0:z(i)/1000:1.6*z(i)],'linewid
th',2)
end

xb=x(2)*1553/1586.5;
yb=y(2)*1553/1586.5;
zb=z(2)*1553/1586.5;

xc=x(3)*1481/1586.5;
yc=y(3)*1481/1586.5;
zc=z(3)*1481/1586.5;
dd=sum(([x(1),y(1),z(1)]-[xc,yc,zc]).^2).^0.5;

plot3([x(1),xb],[y(1),yb],[z(1),zb],'r','linewidth',2)
plot3([xb,xc],[yb,yc],[zb,zc],'r','linewidth',2)

XD=[0:x(5)/10000:2*x(5)];
YD=[0:y(5)/10000:2*y(5)];
ZD=[0:z(5)/10000:2*z(5)];
```

```
r_CD=((xc-XD).^2+(yc-YD).^2+(zc-ZD).^2).^0.5;%取多组D，CD间的距离

dr=abs(r_CD-dd);
I=find(dr==min(dr));

xd=XD(I);yd=YD(I);zd=ZD(I);
plot3([xc,xd],[yc,yd],[zc,zd],'r','linewidth',2)

XE=[0:x(4)/10000:2*x(4)];
YE=[0:y(4)/10000:2*y(4)];
ZE=[0:z(4)/10000:2*z(4)];

r_AE=((x(1)-XE).^2+(y(1)-YE).^2+(z(1)-ZE).^2).^0.5;

dr=abs(r_AE-dd);
I=find(dr==min(dr));

xe=XE(I);ye=YE(I);ze=ZE(I);
plot3([x(1),xe],[y(1),ye],[z(1),ze],'r','linewidth',2)

plot3([xd,xe],[yd,ye],[zd,ze],'r','linewidth',2)

fill3(3.78*100/dd*[xd,xe,x(1),xc],3.78*100/dd*[yd,ye,y(1),yc],3.78*100/dd*[zd,ze,z(
1),zc],'r');
```