数码相机定位

摘要

本问题旨在通过分析靶标和其像的关系，获得两个数码相机之间的距离。由于相机的内置参数各不相同，因此无法通过凸透镜成像的性质及公式获得像物之间的关系。但是在一般情况下，光线都是近轴光且相机对焦准确，那么由经过光心光线不偏转这一事实，可以将相机拍摄过程简化为小孔成像，也就是说物与像之间是中心透视关系。这样，靶标中圆形就透视变换为椭圆，透视变换还有保持切点、公切线和直线的性质。

靶标中，对于某圆，分别求得它与另两个等大的圆的公切线（这里要求三圆心不共线），可得两对在该圆上的切点，分别连接这两对切点，交点即为圆心。由此性质以及前述透视变换的性质，设法对靶标照片上的椭圆重复上述操作，那么两条切点连线的交点即为圆心在像平面上的像。

本文中采用实际实验的检验方法，验证模型的准确性。具体地说，实际地去做一个特殊的靶标（在靶标圆心处附近有同心圆环），并用数码相机拍摄，用之前建立的模型去计算圆心像的位置，再通过这个靶标分析模型的准确性。最后通过不同角度的拍摄，分析模型的稳定性。

为了解决单个相机相对靶标的位置的问题，先选取共线的三个圆心的像，由已经计算得到的圆心像的坐标以及靶标实际尺寸，计算出对应的圆心坐标。通过连线的交点构造另外几组共线的像点，重复上述过程，获得另一组圆心坐标。考虑误差因素，一般地，计算出的圆心的坐标不在同一平面内，故需要利用计算出的点，拟合出一个平面方程，得到靶标平面。最后在靶标上建立一个固定的坐标系，通过变换坐标系，得到光心在靶标坐标系下的坐标。对另一相机重复上述过程，得到在相同坐标系（即靶标坐标系）下该相机光心的位置。由两个相机在靶标坐标系下的位置，可以知道两个相机的相对位置。

最后，本文对模型优缺点、改进方案和相机标定问题的应用前景进行了进一步讨论。

目　录

§1 问题的提出

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。

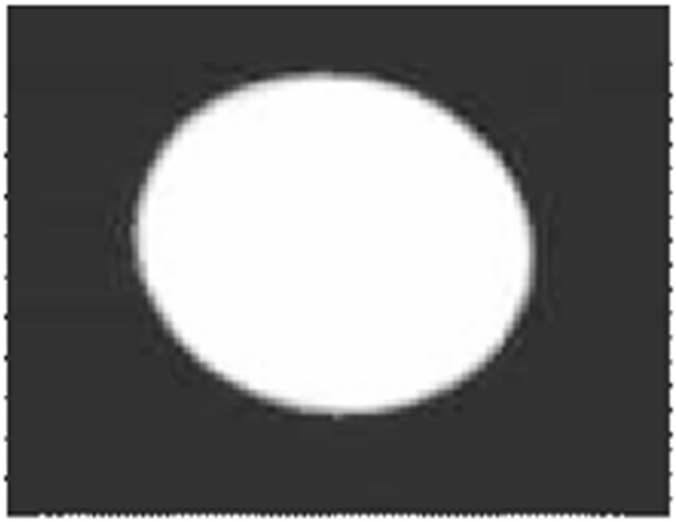

图 1: 靶标上圆的像

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。

用一位置固定的数码相机摄得其像，如图 3 所示。

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心，$x-y$平面平行于像平面；

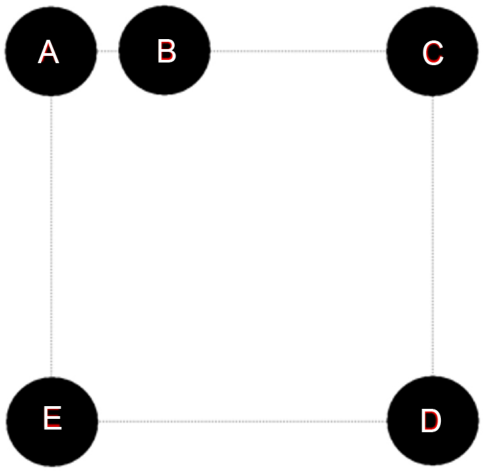

图 2: 靶标示意图

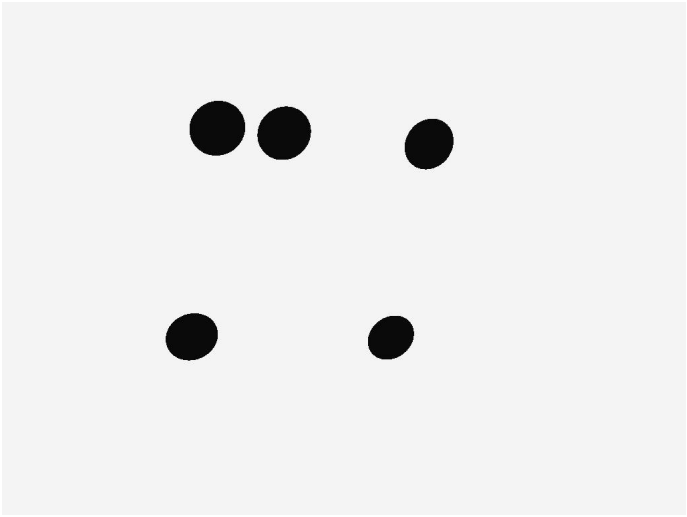

图 3: 靶标的像

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。

§2 问题分析

2.1 数码相机的成像原理

由于过光心的光线不发生折射，而这些光线最终成像于数码相机的电荷耦合器件（又称 CCD，相当于过去相机的胶片），所以我们可以认为在对焦准确后，整个数码相机的拍摄过程就相当于小孔成像，也就是说这是一个透视变换，如图 4 所示。

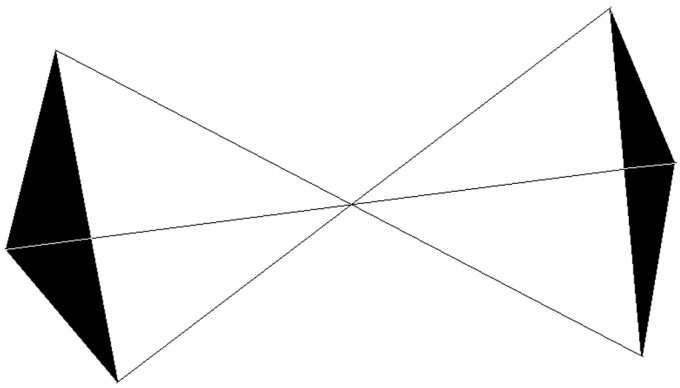

图 4: 透视图

我们称左右两边物体相对于中心点成透视关系，中心点称为透视中心。由上面分析，在数码相机成像过程中，光心就是透视中心，而靶标和靶标的像成透视关系，如图 5 所示。

2.2 问题 1 的分析

文 [1] (P95-P198)中证明了这种透视变换其实是物平面到像平面的一种特殊的射影变换，透视变换保持相切的性质，即变换前相切的两条曲线，在变换后仍然相切，另一方面，透视变换将直线变为直线，将圆变为椭圆。由这些性质，就可以解决第一个问题。

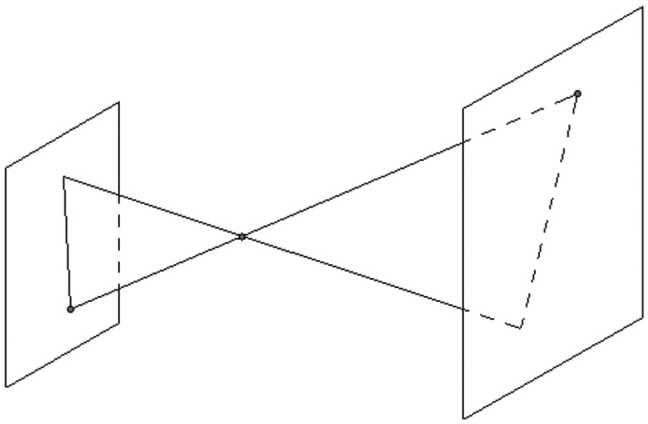

图 5: 数码相机成像原理简化图

2.3 问题 2 的分析

问题 2 是问题 1 建立的数学模型和方法的应用。

2.4 问题 3 的分析

采用实际实验的检验方法，验证模型的准确性。最后通过不同角度的拍摄，分析模型的稳定性。

2.5 问题 4 的分析

通过问题2的答案，即五个圆心像的位置，通过立体几何计算出单个相机相对于靶标的空间位置，进一步确定两相机之间的相对位置。

§3 符号约定

A,B,C,D,E 靶标上的圆，从左上角顺时针方向标记。

A',B',C',D',E' 对应靶标像上的椭圆

O 光心

O_1 光心在靶标像上的投影，即像的中心。

O_A, O_B, O_C, O_D, O_E 各个靶标的圆心。

$O'_A, O'_B, O'_C, O'_D, O'_E$ 各个靶标圆心的像。

§4 模型假设

1. 假设相机成像清晰，且正好成在 CCD 上。

2. 所有光线都是近轴光线，即相机是对着靶标（而不是偏离很远）拍摄的。

由这两个假设，我们可以认为在对焦准确后，整个数码相机的拍摄过程就相当于小孔成像

§5 问题 1

5.1 模型概览

根据模型假设，数码相机的拍照过程模型可以认为是小孔成像，则像对于实物相当于进行了一次透视变换。在 2.2 已经指出，在这种几何变换下，圆的像为椭圆，同时，公切线的相仍为公切线，切点的像仍为切点，直线的像仍为直线。

对于原图中两圆A、B，它们的圆心为O_A, O_B，像为椭圆A', B'。问题 1 要求找到O_A, O_B在A', B'上的像O'_A, O'_B。由于原图中圆A和B等大，则它们的外公切线平行，对于A圆，两条外公切线与其交点（即两个切点）的连线通过圆心O_A，如图 ?? 所示。

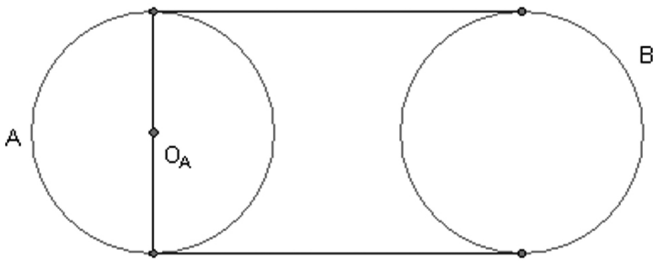

图 6: 物平面上的作图

对于A', B'，也可作其外公切线，两条外公切线与椭圆A'的两个交点的连线记为l_1，由前面论述的透视变换的性质，可以知道l_1通过圆心O_A的像O'_A。如图 ?? 所示。

考虑第三个圆D和其像D'，重复上述过程，可得另一条过O'_A的线段l_2，那么l_1与l_2的交点即为O'_A。如图 8 所示

算法框架如下：

1. 取出像上椭圆边界点，获得此边框上各点坐标。

2. 对这些点进行多元回归拟合，得到椭圆方程。

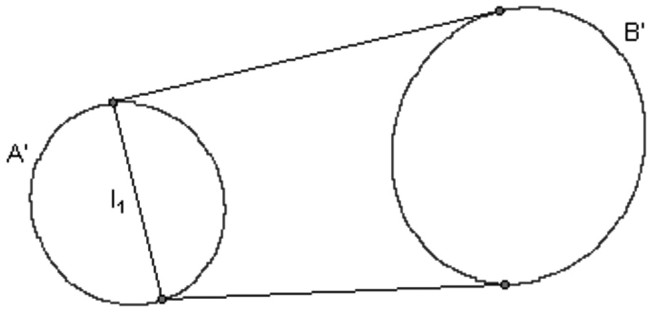

图 7: 对应像平面上的作图

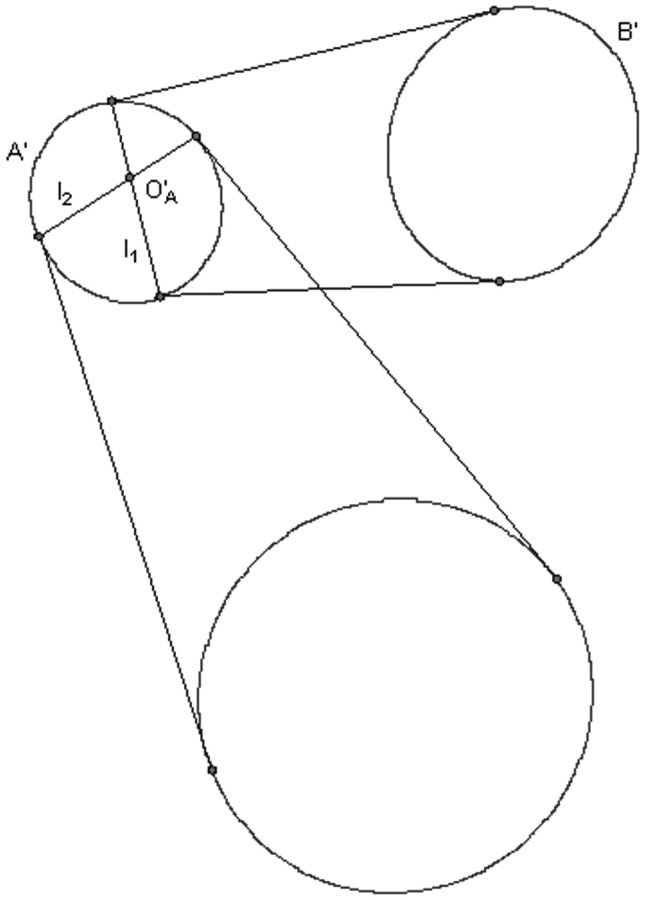

图 8: 像平面上最终的作图

3. 取定一个椭圆，再取另一个椭圆，对于这两个椭圆求出外公切线在某个椭圆上的的切点坐标，计算切点连线的方程，这条连线过圆心的像。

4. 取第三个椭圆，计算出另一条过圆心像的连线方程。

5. 求出以上两条连线的交点，此坐标即为圆心像的坐标。

其中需要给出具体算法的是：分离椭圆边界点，拟合椭圆方程，求两个椭圆公切线与其中一个椭圆的两个切点。

5.2 椭圆边界

如图9在像平面建立直角坐标系，使得左上角的坐标为 $(1,1)$，右上角的坐标为 $(1,1024)$，左下角的坐标为 $(768,1)$，右下角的坐标为 $(768,1024)$。

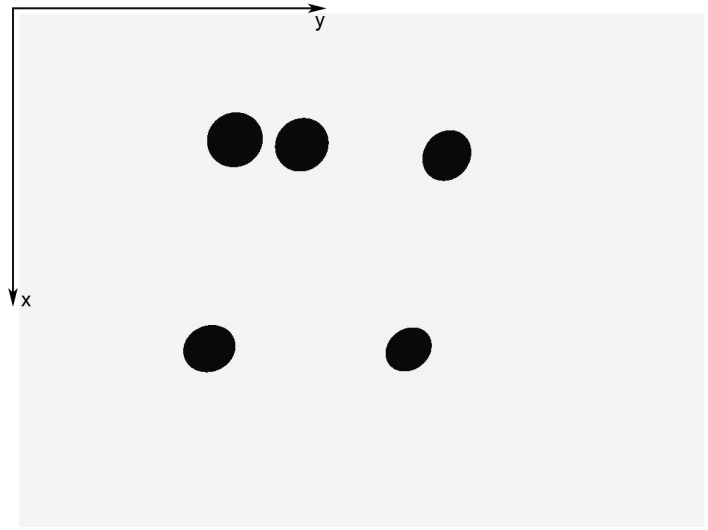

图 9: 坐标系

将图像转换为的矩阵，矩阵每个元素是 0 至 255 的整数，如果一个元素的值与四周（即上下左右）中至少一个元素的差大于 64，那么就认为这个元素是椭圆边界上的元素。由此可以得到椭圆边界。

通过编程将属于某一特定椭圆的边界点的坐标取出。

5.3 拟合椭圆方程

设椭圆方程是

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0 \tag{5.1}$$

由椭圆性质知，有

$$AC > B^2 \geq 0 \tag{5.2}$$

于是 A 非零，可以不妨设$A = 1$，此时椭圆方程是

$$x^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0 \tag{5.3}$$

现在已经知道椭圆边界上的点(x_i, y_i)，我们需要求B, C, D, E, F，使得和

$$\sum \left(x_i^2 + 2Bx_iy_i + Cy_i^2 + 2Dx_i + 2Ey_i + F\right)^2 \tag{5.4}$$

最小，这由最小二乘法即可解决。换言之，只需进行多元回归计算，即可得到最佳的B, C, D, E, F，于是这个椭圆的方程确定了。

5.4 求切点坐标

已知两个相离的椭圆A', B'，在椭圆A'上取点A_1，从A_1向椭圆B'作切线，取其中一个适当的切点B_1，再从B_1向椭圆A'作切线，取适当切点A_2，如此循环，得到收敛的点列$\{A_i\}$。设其极限为A_0，则A_0就是切点。

考虑到透视变换有保公切线的性质，为证明以上事实，其实只需证明两个相离的椭圆变换前的情况（即两个等大的圆）即可。在这种情况下，如图 13 所示，显然有

$$\angle A_1O_AO_B = \angle A_1'O_BO_A < \angle B_1O_BO_A = \angle B_1'O_AO_B < \angle A_2O_AO_B$$

故

$$\angle A_1O_AO_B < \angle A_2O_AO_B$$

于是点列$\{A_i\}$收敛，且只能收敛于切点。

对应的两个椭圆的情况的图解，如图11所示。

需要补充的是计算和选取适当切点的方法。

若已知点A_1坐标(x_1, y_1)，椭圆方程为

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0$$

文[1](P167)指出，从A_1向椭圆作切线的得到的两个切点的连线方程，是

$$Ax_1x + B(x_1y + y_1x) + Cy_1y + D(x_1 + x) + E(y_1 + y) + F = 0 \tag{5.5}$$

连立直线方程和椭圆方程，即可求得两个切点B_1, B_1'的坐标。通过向量积$\overrightarrow{A_1B_1} \times \overrightarrow{A_1B_1'}$的正负来判断切点的取舍。

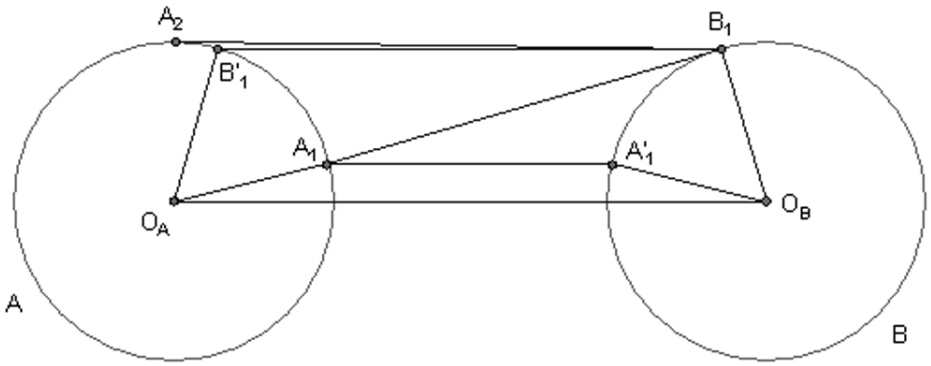

图 10: 收敛的切点

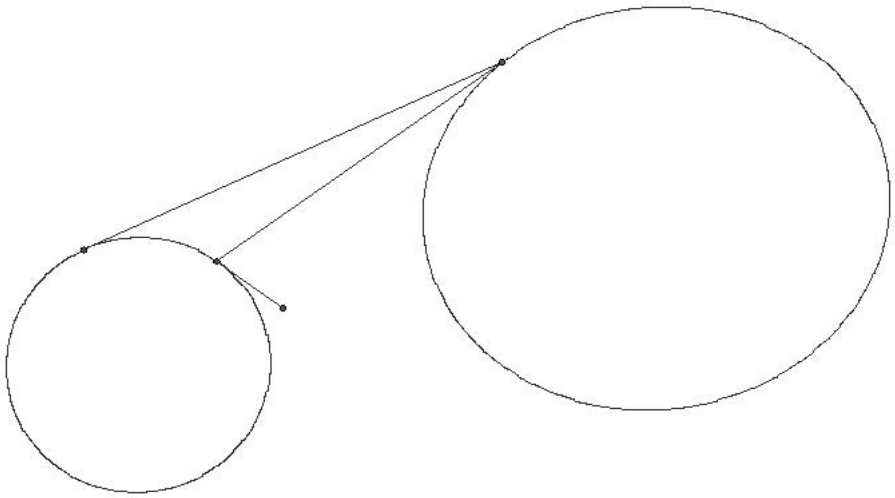

图 11: 椭圆的一般情况

§6 问题2

为了解决问题2，我们首先要考虑得到5个椭圆的外部轮廓：利用MATLAB把题目中的BMP格式图转化为元素为0到255之间整数值的矩阵，扫描每一个点与周围四个点的大小关系，这里我们约定，当周围四点存在一点与该点数值相差大于64时，就认为该点为边界点，具体代码见附录A。通过该程序可求出椭圆的外部轮廓。如图12所示。

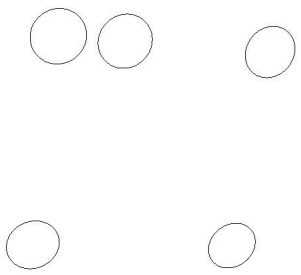

图 12: 椭圆边界

通过编程可以进一步把各个椭圆的边界点区分开（对于问题2，只需要将各个椭圆用矩形范围分开即可），具体代码见附录B。

点的坐标如表 1 所示（共 237 个边界点，只列出部分点的坐标）。

x	148	148	148	148	148	148	148	148	148	148	$\cdots$
y	318	319	320	321	322	323	324	325	326	327	$\cdots$

表 1: 椭圆A'边界点坐标

对于椭圆A'通过多元回归，拟合出边界的曲线方程的系数，具体代码见附录C。对于其他椭圆同样操作，可以得到椭圆方程

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0 \tag{6.1}$$

中的各个系数，如表 2 所示。

通过已知两个椭圆的方程可以计算出外公切线与两椭圆切点的坐标，具体代码见附录D。

	A’	B’	C’	D’	E’
A	1	1	1	1	1
B	0.0429812	0.0687788	0.1356319	0.1690766	0.0930395
C	0.967421	0.9952775	1.0694605	0.906185	0.81669
D	-203.3697	-226.0334	-299.9393	-601.5119	-528.2592
E	-320.52	-434.5436	-713.2621	-613.1145	-279.1671
F	140305.6	226692.4	518901.5	658758.9	343268.3

表 2: 各个椭圆的方程系数

6.1 圆 A 圆心的像

如果取圆A', C'，那么两条外公切线与椭圆A'的切点的坐标为

$$(148.814624, 328.092964), (229.782920, 318.231431)$$

切点连线的方程为

$$9.861533x + 80.968296y - 28032.66855 = 0$$

如果取圆A', E'，那么两条外公切线与椭圆A'的切点的坐标为

$$(192.637470, 363.854745), (186.420279, 281.893064)$$

切点连线的方程为

$$81.961681x - 6.217191y - 13526.73642 = 0$$

则两条切点连线的交点，也就是所要求的圆A的圆心O_A的像O'_A的坐标：

$$(189.5484363, 323.1317896)$$

以上求圆A中心的像O'_A的整个过程如图13所示。

最后，如图14，转换坐标系，求得O'_A在新坐标系中的坐标（仍以1像素作为单位长度）：

$$(-194.9515637, -189.3682104, -1577)$$

6.2 圆 B 圆心的像

如果取圆B', D'，那么两条外公切线与椭圆B'的切点的坐标为

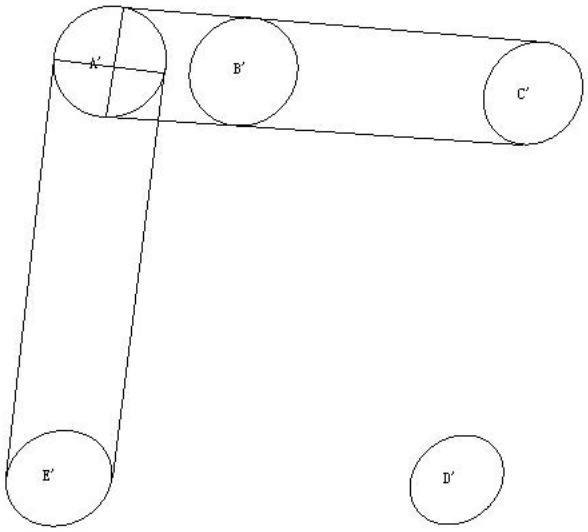

图 13: 示意图

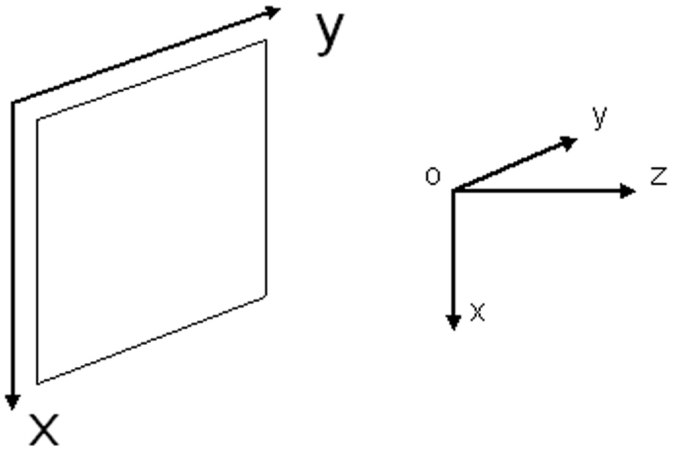

图 14: 转换坐标系，O是光心

$$(217.988429, 386.935508), (176.772415, 459.515746)$$

切点连线的方程为

$$-72.580238x - 41.216014y + 31769.59137 = 0$$

如果取圆B', E'，那么两条外公切线与椭圆B'的切点的坐标为

$$(211.71976, 459.685936), (182.629438, 386.093105)$$

切点连线的方程为

$$73.592831x - 29.090322y - 2208.64462 = 0$$

由此可以计算出两条切点连线的交点，也就是所要求的圆B的圆心O_B的像O'_B的坐标：

$$(197.3374029, 423.3013825)$$

转换坐标系后，得新坐标

$$(-187.1625971, -89.19861752, -1577)$$

6.3 圆 C 圆心的像

如果取圆C', A'，那么两条外公切线与椭圆C'的切点的坐标为

$$(174.858759, 647.599129), (251.469059, 632.636414)$$

切点连线的方程为

$$14.962715x + 76.6103y - 52229.12533 = 0$$

如果取圆C', D'，那么两条外公切线与椭圆C'的切点的坐标为

$$(210.661883, 603.482266), (216.292433, 676.196545)$$

切点连线的方程为

$$-72.714279x + 5.63055y + 11920.18986 = 0$$

则两条切点连线的交点，也就是所要求的圆C的圆心O_C的像O'_C的坐标为：

$$(213.493725, 640.0533553)$$

转换坐标系后，得新坐标

$$(-171.006275, 127.5533553, -1577)$$

6.4 圆 D 圆心的像

如果取圆D', B'，那么两条外公切线与椭圆D'的切点的坐标为

$$(522.163602, 550.917095), (484.421666, 614.888327)$$

切点连线的方程为

$$-63.971232x - 37.741936y + 54196.12667 = 0$$

如果取圆D', C'，那么两条外公切线与椭圆D'的切点的坐标为

$$(502.905551, 548.245872), (503.597421, 617.131732)$$

切点连线的方程为

$$-68.88586x + 0.69187y + 34263.76651 = 0$$

由此可以计算出两条切点连线的交点，也就是所要求的圆D的圆心O_D的像O'_D的坐标：

$$(503.2542872, 582.9677074)$$

转换坐标系后，得新坐标

$$(118.7542872, 70.46770738, -1577)$$

6.5 圆 E 圆心的像

如果取圆E', A'，那么两条外公切线与椭圆E'的切点的坐标为

$$(502.255492, 323.9990714), (501.702376, 245.278890)$$

切点连线的方程为

$$78.7201814x - 0.553116y - 39358.43437 = 0$$

如果取圆E', B'，那么两条外公切线与椭圆E'的切点的坐标为

$$(512.537735, 321.007601), (491.414379, 248.137254)$$

切点连线的方程为

$$72.870347x - 21.123356y - 30568.04477 = 0$$

由此可以计算出两条切点连线的交点，也就是所要求的圆E的圆心O_E的像O'_E的坐标：

$$(501.9785262, 284.5809457)$$

转换坐标系后，得新坐标

$$(117.4785262, -227.9190543, -1577)$$

6.6 问题2的答案

答案见表3。

圆心的像	x	y	z
O'_A	-194.9515637	-189.3682104	-1577
O'_B	-187.1625971	-89.19861752	-1577
O'_C	-171.006275	127.5533553	-1577
O'_D	118.7542872	70.46770738	-1577
O'_E	117.4785262	-227.9190543	-1577

表 3: 各个圆心像的坐标

§7 问题3

7.1 概述

我们通过具体试验的方法，检验模型的正确性和稳定性。我们实际制作一个靶标，靶标上画上3个大小相同的圆，通过之前的模型确定其中一个圆的圆心的像的位置。为了判断正确性，我们事先在其中一个圆的圆心周围标上一些同心圆，这样就可以根据求出圆心的位置在同心圆中的位置判断正确性。最后，通过两个不同角度的拍摄，检验模型的稳定性。

7.2 检验模型具体步骤

首先，我们通过几何画板设计出三个等大的圆，并用Photoshop将其放在一张A4纸上，如图15所示。

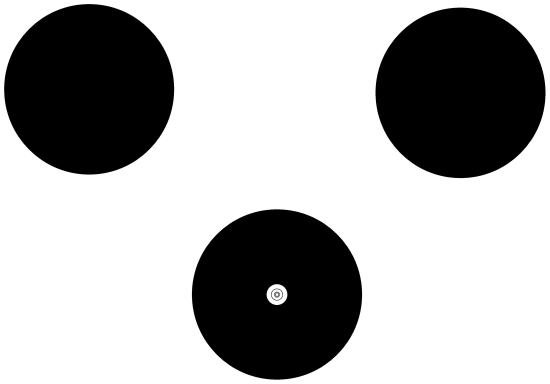

图 15: 靶标设计稿

最下面一个圆中的同心圆半径分别是原来大圆半径的$\frac{1}{8}$，$\frac{1}{16}$，$\frac{1}{32}$，$\frac{1}{64}$，并将其打印出来。

7.3 正面拍摄的检验试验

用Sony T-100相机进行拍摄，分辨率为2048×1536，通过Photoshop将其处理为灰度图，如图16所示。

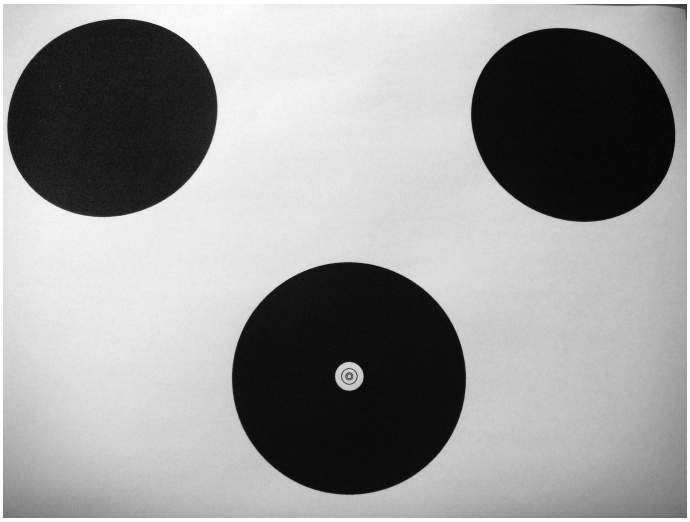

图 16: 靶标照片

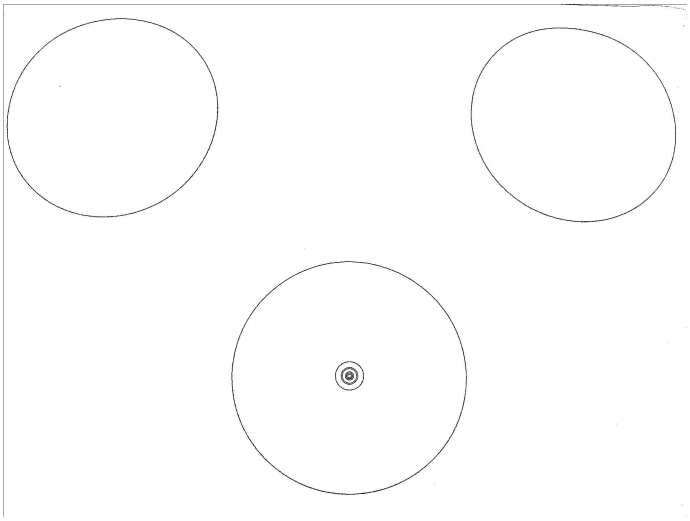

图 17: 边界点

通过之前的算法将边界点寻找出来，如图17所示。

由于背景的问题和光线的问题，需要将一部分杂点（明显不属于椭圆上的点）除去，并将用来判断准确性的同心圆擦去，处理后如图18所示。

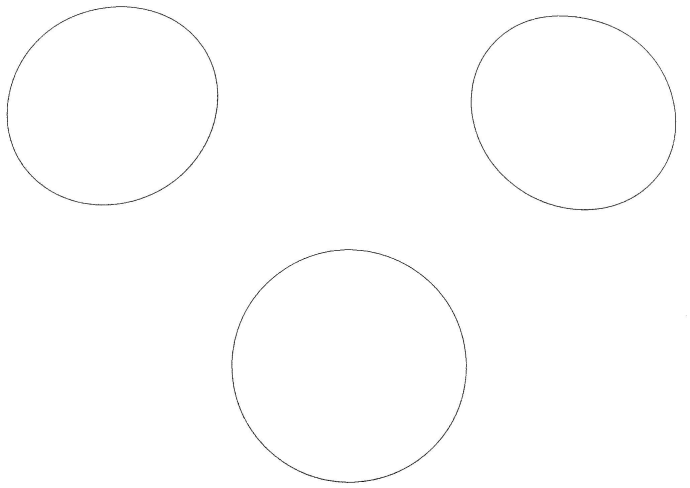

图 18: 去杂后的边界点

坐标系与问题2的坐标类似，左上角的坐标为 $(1,1)$，右上角的坐标为 $(1,2048)$，左下角的坐标为 $(1536,1)$，右下角的坐标为 $(1536,2048)$。上面两个椭圆从左到右记为 F' 和 G'，下面一个圆记为 H'，通过模型我们得到三个圆的边界点的坐标，并拟和出椭圆方程，如表5所示。

如果取圆H',F'，那么两条外公切线与椭圆H'的切点的坐标为

	F'	G'	H'
A	1	1	1
B	0.0784005	-0.1001163	-0.0021277
C	0.886404	0.8980602	0.9818381
D	-365.9928	-189.8709	-1117.6974
E	-316.9231	-1497.8045	-1015.3087
F	140598.5	2543183.1	2183294.6

表 4: 各个椭圆的方程系数

$$(1345.137472, 769.856925), (880.378811, 1290.170837)$$

切点连线的方程为

$$-520.313912x - 464.758661y + 1057691.414 = 0$$

如果取圆H', G'，那么两条外公切线与椭圆H'的切点的坐标为

$$(1340.411320, 1308.128317), (881.649552, 780.613534)$$

切点连线的方程为

$$527.514783x - 458.761768y - 106967.5271 = 0$$

由此可以计算出两条切点连线的交点，也就是所要求的圆的像O'_H的坐标：

$$(1105.556388, 1038.076501)$$

将这个点在原照片中标出，并放大，如图19所示。
这个点位于$\frac{1}{32}$圆内，在$\frac{1}{64}$圆附近。

7.4 倾斜角度拍摄的检验实验

接下来我们换一个稍倾斜的角度拍摄，照片如图20所示。
查找边界后，如图21所示。
去处杂点后，如图22所示。
拟合得椭圆方程系数，如表5所示。
如果取圆H', F'，那么两条外公切线与椭圆H'的切点的坐标为

$$(723.673635, 1432.841295), (1222.712243, 907.159021)$$

切点连线的方程为

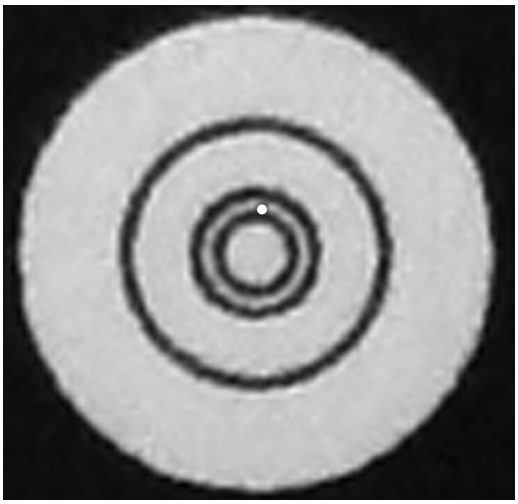

图 19: 求得圆心像的位置

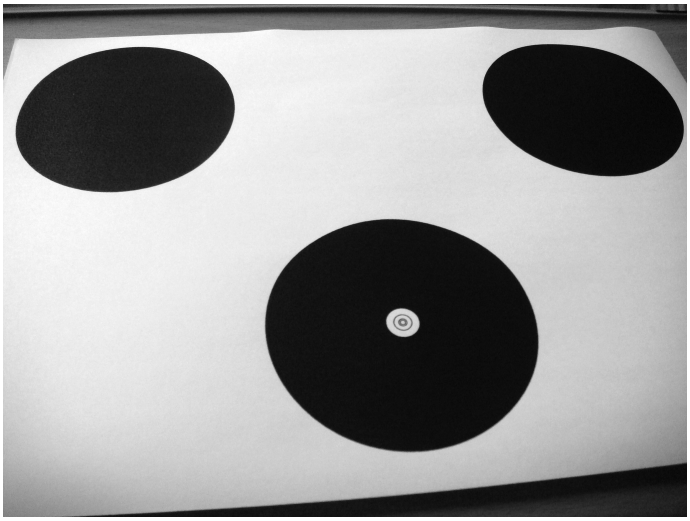

图 20: 靶标照片

	F’	G’	H’
A	1	1	1
B	0.0981824	-0.183513	-0.0587945
C	0.4424182	0.437555	0.722986
D	-381.9474	1.7421	-921.9923
E	-195.9495	-702.1655	-805.3728
F	156912.3	1182682.3	1756296.5

表 5: 各个椭圆的方程系数

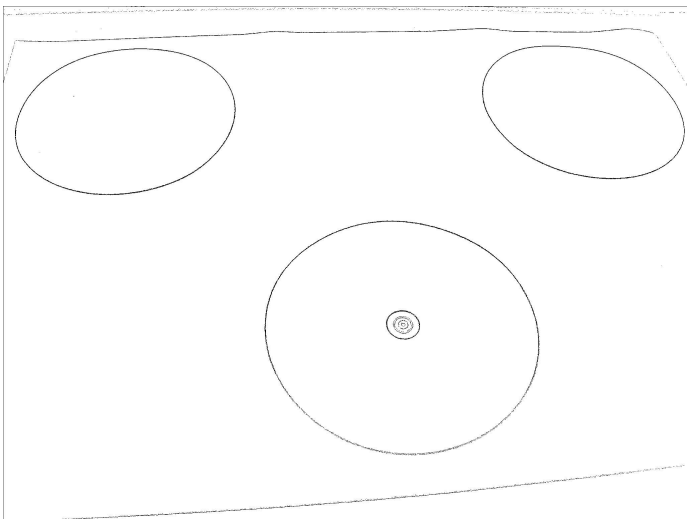

图 21: 边界点

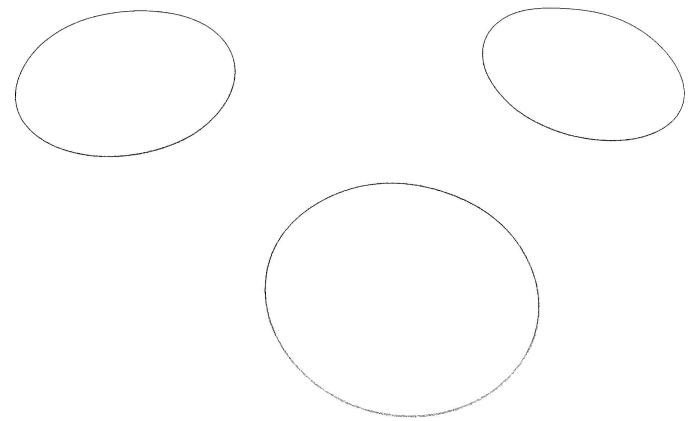

图 22: 边界点

$$525.682274x + 499.038608y - 1095465.527 = 0$$

如果取圆H', G'，那么两条外公切线与椭圆H'的切点的坐标为

$$(752.799192, 878.548222), (1172.416074, 1558.878693)$$

切点连线的方程为

$$-680.330471x + 419.616882y + 143498.5633 = 0$$

由此可以计算出两条切点连线的交点，也就是所要求的圆的像O'_H的坐标：

$$(948.563891, 1195.944151)$$

将这个点在原照片中标出，并放大，如图23所示。

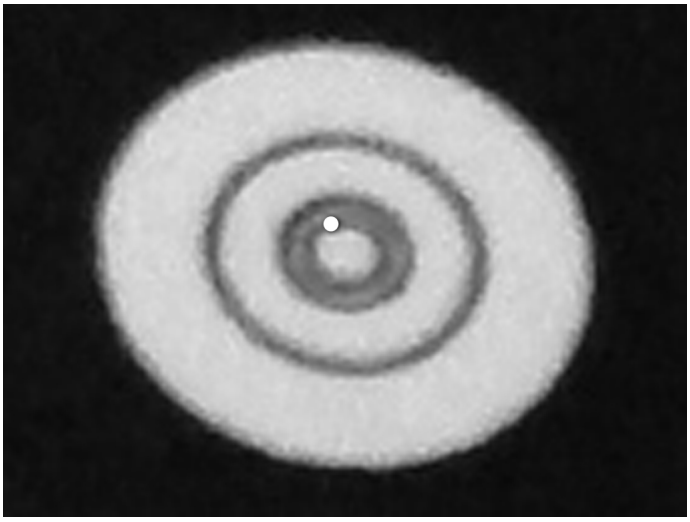

图 23: 求得圆心像的位置

这个点位于$\frac{1}{32}$圆内，在$\frac{1}{64}$圆附近。

7.5 模型准确性与稳定性的讨论

计算椭圆轮廓方程时，由于离散取点并且采用二元回归拟合，因此会造成系统误差，要对模型进行必要的诊断，测试模型假设是否近似成立。通过R软件编程获得残差图，如图24所示。

根据残差图可以知道：

1. 左上图中横轴是对各个观测的拟合值$\widehat{y_i}$，而纵轴是分离出来的残差$\widehat{\varepsilon_i} = y_i - \widehat{y_i}$。从图中看到分布杂乱无章，残差值非常小。说明常方差假设成立。

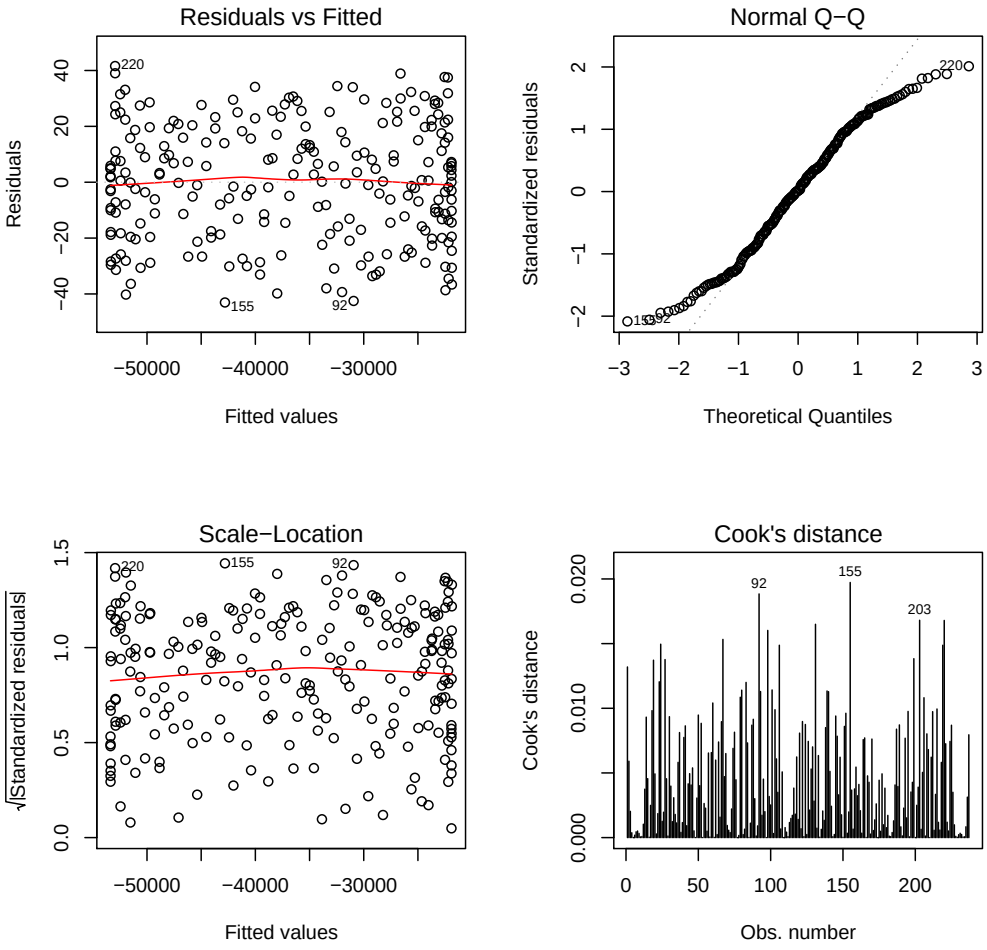
Residuals vs Fitted
Residuals
Fitted values
220
155
92
Normal Q−Q
Standardized residuals
Theoretical Quantiles
220
Scale−Location
√|Standardized residuals|
Fitted values
220
155
92
Cook's distance
Cook's distance
Obs. number
92
155
203

图 24: 残差图

2. 由右上图检验残差的正态性。得到散点图近似直线，说明正态假设成立。

综上，我们认为椭圆的轨迹是相当可靠的，之前的两个实验都证明了这点。

对于来回作切线求切点的过程，程序显示切点序列的坐标在double型变量的精确范围内不变时，就认为已经收敛到了所要求的切点的坐标（该方法收敛速度相当快，一般在做10次切线以内就能收敛到切点坐标），即精度为10^{-6}，可认为该过程造成的误差很小。

对稳定性分析如下：

1. 如果靶标色差过小，即靶标在0到255灰度图片中像点灰度值差距过小，这种情况下无法通过模型获取靶标边界点的方式获得轮廓曲线。这个问题可以通过图形软件进行人工处理解决，事实上在两次试验中我们都对图像的对比度和亮度进行了调整。

2. 由于拍摄时背景和光线影响及纸张自身褶皱造成的阴影，都有可能导致生成灰度图像后，被程序判定为边界点，即成为拟合曲线方程时的杂点，这时可以人工去除这些杂点。

3. 如果靶标设计采用空心圆，若边框线条过细，那么通过模型中的程序处理后，重新生成的图像边框可能模糊或者间断。但此时我们的模型仍然适用。由于只需获得大量离散点，就能拟合出椭圆方程，因此这个过程与重新生成的图像是否连续无关。

4. 由实验结果表明，正面拍摄与斜面拍射对于圆心像坐标的精度没有显著影响。

5. 如果靶标放置时，表面有弯曲，这样所拍摄到的图像就不能认为是椭圆，因为靶标自身已经发生了变形，这样造成的误差是无法修正的，我们只能尽量将纸放平，减小这种情况造成的影响。

§8 问题4

8.1 模型概览

在问题4中，我们首先解决一个相机相对于靶标的位置，具体地，我们通过三个共线的点像的位置，并结合光心的位置，以及原来靶标的尺寸，这些参数确定这三个共线点原像的位置。通过这种算法，考虑多个共线三点组，求得它们原像的坐标，并拟合出一个平面，这就是靶标平面。通过坐标变换，就能确定单个相机相对于靶标的位置。最后利用两个相机相对于靶标的位置解决这两个相机相对位置的问题。

注意！在问题4中，我们已经不考虑椭圆和圆，取而代之的是圆中心的像和圆心，在记号上A就表示圆A的圆心，其他类似。

8.2 求共线三点原像坐标

已知共线三点的像在坐标系（以光心为原点的那个坐标系）中的位置和原来靶标上这三点的之间的距离（换算为以像素为单位），我们利用以下算法求出这三点在靶标上的坐标，具体代码见附录E。

如图25所示，已知A’、B’、C’三点的坐标(x_1, y_1, z_1)，(x_2, y_2, z_2)，(x_3, y_3, z_3)，光心坐标$(0, 0, 0)$以及AB的长度和BC的长度，下面计算A、B、C三点的坐标。

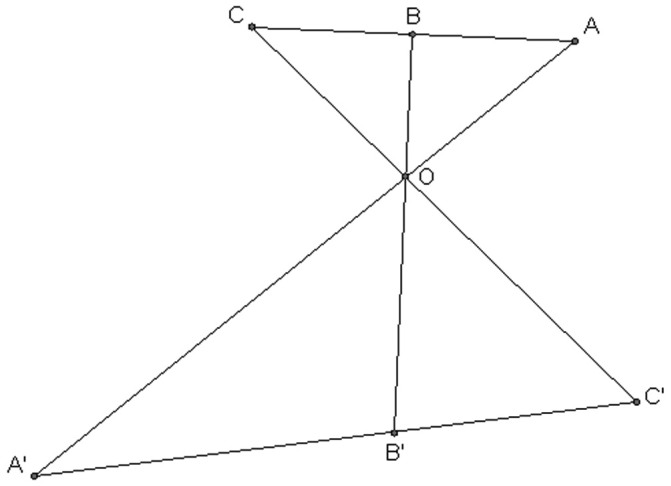

图 25: 图示

利用两点距离公式求出$A'B'$和$B'C'$的距离，OA'的距离，OC'的距离，于是，可得

$$
\begin{aligned}
1 &= \frac{S_{A'OB'}}{S_{B'OC'}} \times \frac{S_{B'OC'}}{S_{BOC}} \times \frac{S_{BOC}}{S_{AOB}} \times \frac{S_{AOB}}{S_{A'OB'}} && (8.1)\\
&= \frac{A'B'}{B'C'} \times \frac{B'O \times C'O}{BO \times CO} \times \frac{BC}{AB} \times \frac{AO \times BO}{A'O \times B'O} && (8.2)\\
&= \frac{A'B'}{B'C'} \times \frac{C'O}{CO} \times \frac{BC}{AB} \times \frac{AO}{A'O} && (8.3)
\end{aligned}
$$

这样就能求出AO/CO的值：

$$\frac{AO}{CO} = \frac{B'C'}{A'B'} \times \frac{A'O}{C'O} \times \frac{AB}{BC}$$

又因为

$$\frac{AO^2 + CO^2 - AC^2}{2AO} = \cos\angle AOC = \cos\angle A'OC' = \frac{\overrightarrow{A'O} \cdot \overrightarrow{C'O}}{\left|\overrightarrow{A'O}\right| \left|\overrightarrow{C'O}\right|}$$

联立上面两个方程，就可以求出AO, CO的长度，从而利用定比分点坐标公式求出A, B, C的坐标。

我们在像上取共线的三点组，

8.3 单相机相对靶标位置

对于$A'B'C'$通过以上步骤可以得到

原像A坐标：

$$(210.665108, 204.631724, 1704.109828)$$

原像B坐标：

$$(207.016607, 98.660712, 1744.286486)$$

原像C坐标：

$$(199.311991, -148.666551, 1838.032021)$$

下面我们构造更多的共线的点，先求出$A'D'$和$C'E'$的交点F',如图26所示。

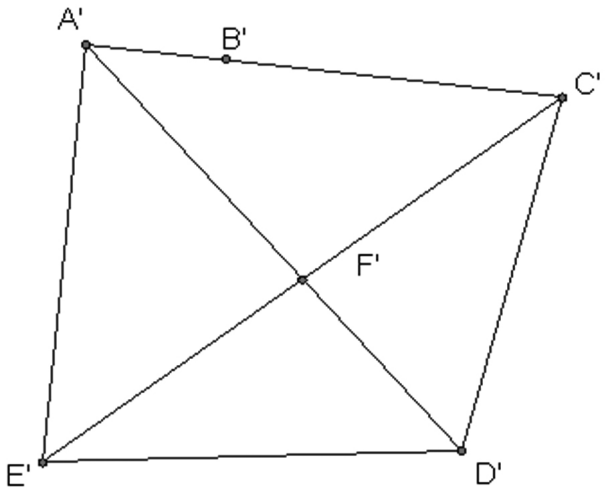

图 26: 图示

对应靶标上有F，如图27，这里再次用到

对于直线$A'D'$，其方程可以计算得到为：

$$-259.8359178x + 313.7058509y + 8750.4971 = 0$$

直线$C'E'$，方程为：

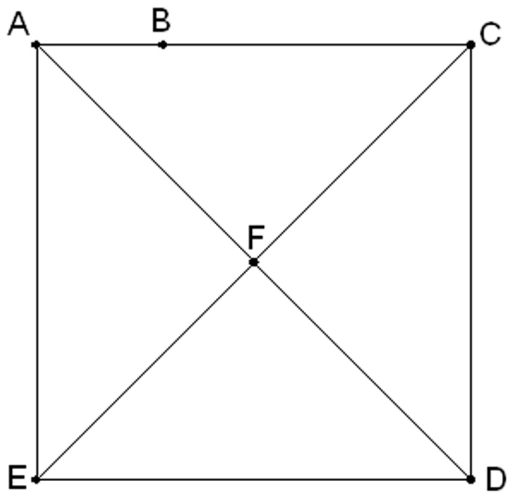

图 27: 图示

$$355.4724096x + 288.4848012y + 23990.80828 = 0$$

这样就可以求出交点F'为

$$(-26.82257082, -50.11052348, -1577)$$

对于共线三点$A'F'D'$，重复以上过程，得到
原像A坐标

$$(211.289539, 205.238272, 1709.160970)$$

原像F坐标

$$(31.322178, 58.516790, 1841.548869)$$

原像D坐标

$$(-148.645183, -88.204692, 1973.936767)$$

对于共线三点$C'F'E'$，可以得到
原像C坐标

$$(199.770662, -149.008674, 1842.261837)$$

原像F坐标

$$(31.321559, 58.515633, 1841.512472)$$

原像E坐标

$$(-137.127544, 266.039941, 1840.763107)$$

对于多次计算的点（即点A和点C）我们取其平均数，于是就可以得到
原像A坐标

$$(210.9773235, 204.934998, 1706.635399)$$

原像B坐标

$$(207.016607, 98.660712, 1744.286486)$$

原像C坐标

$$(199.5413265, -148.8376125, 1840.146929)$$

原像D坐标

$$(-148.645183, -88.204692, 1973.936767)$$

原像E坐标

$$(-137.127544, 266.039941, 1840.763107)$$

原像F坐标

$$(31.3218685, 58.5162115, 1841.530671)$$

8.4 拟合靶标平面

以上求出的原像应当位于一个平面上，但由于各种误差，我们需要去拟合一个平面：

$$Ax + By + Cz + D = 0$$

使得

$$\sum (x_i + By_i + Cz + D)^2$$

最小，这里已经不妨设$A = 1$。

得到最终拟合的系数：

$$A = 1, B = 0.8048061, C = 2.2160054, D = -4156.1877$$

这样靶标所在平面方程就是

$$x + 0.8048061y + 2.2160054z - 4156.1877 = 0$$

再通过射线$A'O$与靶标平面的交点确定A在靶标平面上的坐标：

$$(210.8943187, 204.8543697, 1705.963954)$$

同理可以得到C在靶标平面上的坐标：

$$(199.4772667, -148.7898305, 1839.556177)$$

我们将这两个坐标作为A, C最终的坐标。

8.5 求解光心相对于靶标坐标系的位置

此时建立新的坐标系，以A

$$(210.8943187, 204.8543697, 1705.963954)$$

为原点，$\overrightarrow{AC}$的单位向量

$$(-0.030188495, -0.935054967, 0.35321504)$$

作为x方向，平面的单位法向量

$$(-0.390482242, -0.31426249, -0.865310757)$$

作为z方向（这个法向量使得O在这个法向量所指的半空间中），y方向由向量积

$$\overrightarrow{z} \times \overrightarrow{x}$$

确定，为

$$(0.920115359, -0.16404663, -0.355635248)$$

通过坐标转换公式，可以得出相机光心O在新坐标系下的坐标：

$$(-404.6554483, 446.2594811, 1622.917491)$$

于是在靶标坐标系中，相机的位置已经确定（单位为像素），同样对另一个相机，在靶标坐标系中也有一个坐标，这样两个相机的相对位置就确定了。

§9 数码相机定位问题及其数学模型的进一步讨论

9.1 模型的优点

1. 利用灰度值的差异获得的轮廓曲线较为清晰，对于题目中的像图运用此方法进行轮廓取样，近似不失真。

2. 对轮廓上所有点进行多元拟合，相当于通过增加样本数量减少误差。经过残差图分析可知，拟合效果较为理想。

3. 利用投影变换的性质运用迭代法（来回反复作切线）求切线是本模型的亮点。对于每组数据的收敛速度在10次迭代之内，而且数值精确度与double型变量相同，为10e-6，对于模型整体精确性影响很小。

9.2 模型的缺点

1. 由于数码相机内置参数不同（如焦距不同），故模型无法采用透镜成像原理进行分析，而是模型化为小孔成像，可能造成系统误差。

2. 对于通过切线组相交求得圆心投影的过程，模型中采用连接两组切点，连线交点即为圆心投影的方式。事实上，如果对于同一个投影图形，多次进行上述操作，可获得多组圆心投影坐标，再通过取平均值，应能获得更精确的坐标数据。

9.3 应用前景

对于数码相机定位问题，本质在于如果两相机相对位置已知，通过分析成像结果，运用立体几何运算，能够获得实物的真实尺寸和距离，相当于具有空间距离的感觉，与人眼机理相同。那么，运用这种性质，机器人也可具有判断空间物体形状和距离的能力，即此问题也适用于智能感知等领域。

A 寻找椭圆边界的程序代码

使用 MATLAB 7.4.0 编程环境。

```
clear all;
a = imread('a.bmp', 'bmp');
a = a(:,:,1);
e = 64;
for i = 2:767
    for j = 2:1023
        if abs(a(i, j) - a(i - 1, j)) > e
        || abs(a(i, j) - a(i + 1, j)) > e
        || abs(a(i, j) - a(i, j - 1)) > e
        || abs(a(i, j) - a(i, j + 1)) > e
            b(i, j) = 0;
        else
            b(i, j) = 255;
        end
    end
end
b(:,1024) = 255;
b(768,:) = 255;
imwrite(b, 'b.bmp', 'bmp');
```

B 选出特定椭圆边界点的程序代码

使用 MATLAB 7.4.0 编程环境。

对于椭圆 A，代码如下

```
for i = 138:245
    for j = 270:375
        if b(i,j) < 128
            l = l + 1;
            c(l, 1:2) = [i, j];
        end
    end
end
```

对于椭圆 B，代码如下

```
for i = 143:245
    for j = 372:472
        if b(i,j) < 128
            l = l + 1;
            c(l, 1:2) = [i, j];
        end
    end
end
```

对于椭圆 C，代码如下

```
for i = 161:261
    for j = 591:689
        if b(i,j) < 128
            l = l + 1;
            c(l, 1:2) = [i, j];
        end
    end
end
```

对于椭圆 D，代码如下

```
for i = 452:553
    for j = 528:633
        if b(i,j) < 128
            l = l + 1;
            c(l, 1:2) = [i, j];
        end
    end
end
```

对于椭圆 E，代码如下

```
for i = 460:543
    for j = 236:333
        if b(i,j) < 128
            l = l + 1;
            c(l, 1:2) = [i, j];
        end
    end
end
```

C 拟合椭圆方程的程序代码

使用 R 2.5.1 编程环境。

```
rm(list=ls())
a=read.table("D:/a/c1.txt",header=T)
lm1=lm(-xx~xy+yy+x+y,data=a)
summary(lm1)
```

D 求切点坐标的程序代码

使用 XCode 编程环境。

```
#include <stdio.h>
#include <math.h>

typedef struct {
    double x;
    double y;
} point;

int cal(double x, double y, double A, double B, double C, double D,
double E, double F, point *p1, point *p2);
double vector(point p1,point p2, point p3);

int main() {
    int i, f[2] = {1, -1};
    point dp, p[2];
    dp.x = 340;
    dp.y = 355;
    double A[2] = {1, 1}, B[2] = {0.1356319, 0.1690766},
    C[2] = {1.0694605, 0.906185}, D[2] = {-299.9393, -601.5119},
    E[2] = {-713.2621, -613.1145}, F[2] = {518901.5, 658758.9};
    for (i = 0; i < 10; i++)
    {
        cal(dp.x, dp.y, A[i % 2], B[i % 2], C[i % 2],
```

```
        D[i % 2], E[i % 2], F[i % 2], p, p + 1);
        if (vector(dp, p[0], p[1]) / f[i % 2] > 0) dp = p[0];
        else dp = p[1];
        printf("%f %f\n", dp.x, dp.y);
    }
    return 0;
}

int cal(double x1, double y1, double A, double B, double C, double
D, double E, double F, point *p1, point *p2) {
    double a = A * x1 + B * y1 + D,
    b = B * x1 + C * y1 + E, c = D * x1 + E * y1 + F;
    double u = A * b * b - 2 * B * a * b + C * a * a;
    double v = - B * b * c + a * C * c + D * b * b - E * a * b;
    double w = C * c * c - 2 * E * b * c + F * b * b;
    p1->x = (-v + sqrt(v * v - u * w)) / u;
    p1->y = (-a * p1->x - c) / b;
    p2->x = (-v - sqrt(v * v - u * w)) / u;
    p2->y = (-a * p2->x - c) / b;
    return 0;
}

double vector(point p1, point p2, point p3) {
    return (p2.x - p1.x) * (p3.y - p1.y)
    - (p2.y - p1.y) * (p3.x - p1.x);
}
```

E 计算共线三点原像的程序代码

使用 Visual Studio 2005 编程环境。

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
int main() {
    double x1,x2,x3,y1,y2,y3,z1,z2,z3;
```

```
    double r,r1,r2,p1,p2,k,m,l,l1,l2,r3,r4;
    scanf("%lf %lf %lf %lf %lf %lf %lf %lf %lf",
    &x1,&y1,&z1,&x2,&y2,&z2,&x3,&y3,&z3);
    scanf("%lf %lf",&r3,&r4);
    m=r3+r4;
    r=r3/r4;
    r1=sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2)+(z1-z2)*(z1-z2));
    r2=sqrt((x2-x3)*(x2-x3)+(y2-y3)*(y2-y3)+(z2-z3)*(z2-z3));
    p1=sqrt(x1*x1+y1*y1+z1*z1);
    p2=sqrt(x3*x3+y3*y3+z3*z3);
    k=r1*p2/p1/r2/r;
    l=m/sqrt(1+k*k-k*((p1*p1+p2*p2-(r1+r2)*(r1+r2))/p1/p2));
    l1=sqrt((r2*p1*p1+p2*p2*r1-(r1+r2)*r1*r2)/(r1+r2));
    l2=sqrt((l*l*r4+k*k*r3*l*l-r3*r3*r4-r3*r4*r4)/(r3+r4));
    printf("%lf %lf %lf\n",-x1*l/p1,-y1*l/p1,-z1*l/p1);
    printf("%lf %lf %lf\n",-x2*l2/l1,-y2*l2/l1,-z2*l2/l1);
    printf("%lf %lf %lf\n",-x3*k*l/p2,-y3*k*l/p2,-z3*k*l/p2);
    return 0;
}
```

参考文献

[1] David A. Brannan，Matthew F. Esplen, Jeremy J. Gray *Geometry*，Cambridge University Press, 1999

[2] 王汉生*应用商务统计分析*, 北京大学出版社, 2008