《计算机视觉与模式识别》

课程实习报告

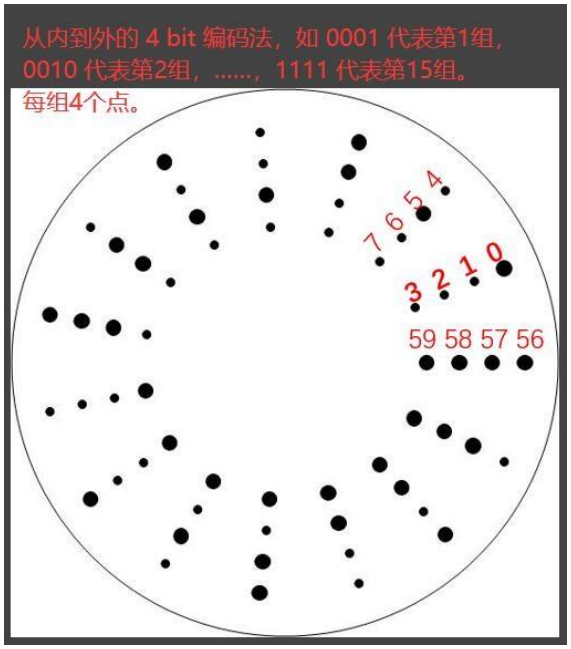
一、基本原理

1. 寻找控制点坐标

控制点就是圆盘上黑色区域的中心坐标，体现在图像上就是黑色像素集合区域的中心，那我们首先要做的就是提取出黑色像素。由于原图像为 RGB 彩色图像，佛像与背景、控制点黑色区域与圆盘其他部分对比度较高，故将图像二值化后，进行提取黑色像素的操作，进而提取出控制点坐标。

2. 对控制点进行编号

提取出控制点坐标后，接下来的任务就是利用已知的控制点编号规则在图像上对控制点进行编号。



注意到控制点是在一个圆盘之上的，即是说：

任意过同一编号组控制点的直线一定都相交于圆盘圆心，圆心的直线至多与 4 个控制点中心相交，又有：投影（相机摄像）不会改变物体的相对位置，并且图片已经消除了畸变，不会出现同一组控制点不在一条直线上的情况。

基于以上事实，我们可以得到将控制点编号的方法：

①选取两组控制点中各两点，列直线方程；

②计算两直线相交点作为圆盘中心点；

③逐点进行遍历并分组：首先任意选取一点作为起始点，然后应用直线拟合对其他点进行判定（即是否为同一组控制点）并记录同一组的控制点的像素坐标，然后不断选取未分组的点进行分组操作，直到全部点分组完毕；

④将分组后的控制点从外到内进行排序：在分好组的控制点间计算其与中心点之间的棋盘距离，采用冒泡排序以便于同时更新控制点坐标；

⑤编号：根据控制点区域大小赋给控制点不同的值（0 或 1），4 个点得到一个二进制数（不满 4 个点的控制点组舍弃），依此序列计算组号。

3、计算相机外部参数

采取后方交会计算左右相片各自的外部参数。

共线条件方程：

$$x - x_0 = -f\frac{a_1(X-X_S)+b_1(Y-Y_S)+c_1(Z-Z_S)}{a_3(X-X_S)+b_3(Y-Y_S)+c_3(Z-Z_S)} = -f\frac{\overline{X}}{\overline{Z}}$$

$$y - y_0 = -f\frac{a_2(X-X_S)+b_2(Y-Y_S)+c_2(Z-Z_S)}{a_3(X-X_S)+b_3(Y-Y_S)+c_3(Z-Z_S)} = -f\frac{\overline{Y}}{\overline{Z}}$$

其中，旋转和平移分别如下：

$$R = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$$

$$= \begin{bmatrix} cos\varphi cos\,\kappa - \sin\phi\sin\omega\sin\kappa & -cos\varphi sin\,\kappa - \sin\phi\sin\omega\cos\kappa & -\sin\phi\cos\omega \\ cos\,\omega\, sin\,\kappa & cos\,\omega\, cos\,\kappa & -sin\,\omega \\ sin\varphi cos\,\kappa + \cos\phi\sin\omega\sin\kappa & -sin\varphi sin\,\kappa + \cos\phi\sin\omega\cos\kappa & \cos\phi\cos\omega \end{bmatrix}$$

$$T = \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix}$$

由共线条件方程，以像点坐标为观测值，可得误差方程：

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial X_S} & \frac{\partial x}{\partial Y_S} & \frac{\partial x}{\partial Z_S} & \frac{\partial x}{\partial \phi} & \frac{\partial x}{\partial \omega} & \frac{\partial x}{\partial \kappa} \\ \frac{\partial y}{\partial X_S} & \frac{\partial y}{\partial Y_S} & \frac{\partial y}{\partial Z_S} & \frac{\partial y}{\partial \phi} & \frac{\partial y}{\partial \omega} & \frac{\partial y}{\partial \kappa} \end{bmatrix} \begin{bmatrix} \Delta X_S \\ \Delta Y_S \\ \Delta Z_S \\ \Delta\phi \\ \Delta\omega \\ \Delta\kappa \end{bmatrix} - \begin{bmatrix} x-(x) \\ y-(y) \end{bmatrix}$$

$$V = Bx^T - L$$

对共线方程线性化后有

$$x^T = [\Delta X_S \quad \Delta Y_S \quad \Delta Z_S \quad \Delta\varphi \quad \Delta\omega \quad \Delta\kappa]$$

$$B = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \end{bmatrix}$$

$$l = \begin{bmatrix} l_x \\ l_y \end{bmatrix}$$

$$\begin{cases} a_{11} = \frac{1}{\overline{Z}}[a_1 f + a_3(x-x_0)] \\ a_{12} = \frac{1}{\overline{Z}}[b_1 f + b_3(x-x_0)] \\ a_{13} = \frac{1}{\overline{Z}}[c_1 f + c_3(x-x_0)] \\ a_{21} = \frac{1}{\overline{Z}}[a_2 f + a_3(y-y_0)] \\ a_{22} = \frac{1}{\overline{Z}}[b_2 f + b_3(y-y_0)] \\ a_{23} = \frac{1}{\overline{Z}}[c_2 f + c_3(y-y_0)] \end{cases} \quad \begin{cases} a_{14} = (y-y_0)\sin\omega - \{\frac{(x-x_0)}{f}[(x-x_0)\cos\kappa - (y-y_0)\sin\kappa] + f\cos\kappa\}\cos\omega \\ a_{15} = -f\sin\kappa - \frac{x-x_0}{f}\{(x-x_0)\sin\kappa + (y-y_0)\cos\kappa\} \\ a_{16} = +(y-y_0) \\ a_{24} = -(x-x_0)\sin\omega - \{\frac{(y-y_0)}{f}[(x-x_0)\cos\kappa - (y-y_0)\sin\kappa] - f\cos\kappa\}\cos\omega \\ a_{25} = -f\sin\kappa - \frac{y-y_0}{f}\{(x-x_0)\sin\kappa + (y-y_0)\cos\kappa\} \\ a_{26} = -(x-x_0) \end{cases}$$

计算 x 的法方程为 $x=(B^TB)^{-1}Bl$

解算的具体流程：

（1） 对给定的图像进行处理，提取控制点的像点坐标并正确编号，和控制点物方坐标一一对应

（2） 对每个控制点列误差方程

（3） 跟据误差方程列法方程

（4） 求解法方程得到改正数ΔX_S ΔY_S ΔZ_S $\Delta\phi$ $\Delta\omega$ $\Delta\kappa$

（5） 更新外方位元素，重复以上步骤，直到满足退出条件为止

$$X_S = X_S + \Delta X_S \quad Y_S = Y_S + \Delta Y_S \quad Z_S = Z_S + \Delta Z_S$$

$$\phi = \phi + \Delta\phi \quad \omega = \omega + \Delta\omega \quad \kappa = \kappa + \Delta\kappa$$

（6） 得到最终的外部参数X_S Y_S Z_S ϕ ω κ

4、用前方交会原理计算控制点三维坐标

参考 CSDN 文章：https://blog.csdn.net/AAAA202012/article/details/124197239

利用共线方程，采用摄影测量前方交会的方式

已知：相机内参：x_0,y_0,f,width,height

相片外参：X_{S1},Y_{S1},Z_{S1},φ_1,ω_1,$\kappa_1$$X_{S2}$,$Y_{S2}$,$Z_{S2}$,$\varphi_2$,$\omega_2$,$\kappa_2$

同名像点坐标：(x1，y1)，(x2，y2)

求：控制点三维坐标：X,Y,Z

由两幅图像的外方位元素φ1，ω1，κ1 和φ2，ω2，κ2 计算相应的旋转正交矩阵 R1,R2

$$R1 = \begin{bmatrix} a_{11} & b_{11} & c_{11} \\ a_{21} & b_{21} & c_{21} \\ a_{31} & b_{31} & c_{31} \end{bmatrix} \qquad R1 = \begin{bmatrix} a_{12} & b_{12} & c_{12} \\ a_{22} & b_{22} & c_{22} \\ a_{32} & b_{32} & c_{32} \end{bmatrix}$$

像点的像空间辅助坐标：

$$\begin{bmatrix} X1 \\ Y1 \\ Z1 \end{bmatrix} = R1 \begin{bmatrix} x1 \\ y1 \\ -f \end{bmatrix} \qquad \begin{bmatrix} X2 \\ Y2 \\ Z2 \end{bmatrix} = R2 \begin{bmatrix} x2 \\ y2 \\ -f \end{bmatrix}$$

基线分量:

$$Bx=X_{S1}-X_{S2}$$

$$By=Y_{S1}-Y_{S2}$$

$$Bz=Z_{S1}-Z_{S2}$$

投影系数:

$$N_1=\frac{BxZ2-BzX2}{X1Z2-ZX2}$$

$$N_2=\frac{BxZ1-BzX1}{X1Z2-ZX2}$$

地面点的地面坐标:

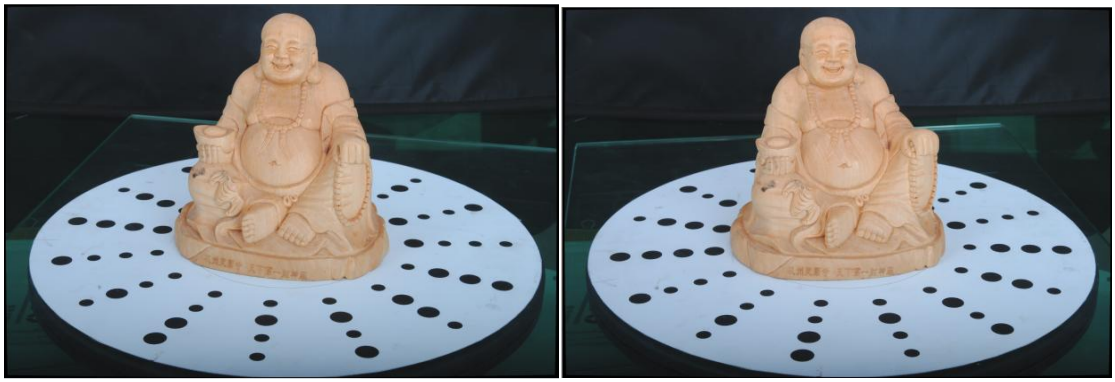
$$X=X_{S1}+N_1X1$$

$$Y=[(Y_{S1}+N_1Y1)+(Y_{S2}+N_2Y2)]/2$$

$$Z=Z_{S1}+N_1Z1$$

二、原始数据

无畸变影像左右各一张:



外部参数：

依次是 Xs Ys Zs Phi Omega Kappa

原始影像-左：

350.0　520.0　300.0

-0.9209585845351669 -0.8780690992255569 2.1102441253730908

原始影像-右：130.0　610.0　300.0

-0.4545695618308865 -1.1219147656947568 2.6531277188826015

控制点坐标：

60　# ID, X, Y, Z, control(1)_or_check(2)

0　132.50000000000000　0.00000000000000　0.00000000000000　1

1　114.00000000000000　0.00000000000000　0.00000000000000　2

2　95.50000000000000　0.00000000000000　0.00000000000000　1

3　77.00000000000000　0.00000000000000　0.00000000000000　2

.....

59　70.34300534935637　-31.31871003761855　0.00000000000000　2

三、实现步骤

1、预处理

在正式对图像进行处理之前，为了提高提取控制点的效率与精度，首先进行二值化处理，简化控制点的提取。

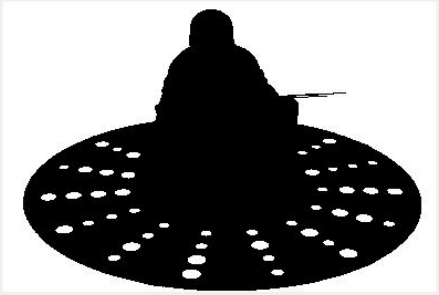
实现代码如下：

```
LeftRgb = imread('无畸变影像-左.bmp');
Gray = rgb2gray(LeftRgb);
t = graythresh(Gray);%计算二值转化的阈值
Gray = imbinarize(Gray, t);%进行二值化
leftBI = Gray;
Gray = ~Gray;
Gray= bwareaopen(Gray,1000);%去除小的杂点
subplot(1,2,1),imshow(LeftRgb);title('无畸变影像-左');
subplot(1,2,2),imshow(Gray);title('二值化处理结果');
```

处理结果如下：

无畸变影像-左

二值化处理结果



无畸变影像-右

二值化处理结果

2、寻找控制点坐标

对二值图取质心及面积，储存于结构体 ldftInfo.coodInfo 中

实现代码如下：

```
[L,num] = bwlabel(Gray);%标记连通域
STATS = regionprops(L,'Centroid','Area');%取质心
data = zeros(num,2); %存储所有连通域的质心
area = zeros(num,1); %存储所有连通域的面积
for i=1:num %遍历
  data(i,:)=STATS(i).Centroid;
  area(i,:)=STATS(i).Area;
end
eVal = Gray(round(data(:,2)), round(data(:,1)))==1; %取有效值,即质心必须落在控制点内
leftInfo.coodInfo = [data(diag(eVal),:) area(diag(eVal),:)]; %左边控制点坐标
```

处理结果如下：

leftInfo.coodInfo

	1 X	2 Y	3 面积
1	409.2966	1.7470e+03	6852
2	548.6247	2.0378e+03	8628
3	546.3453	1.4805e+03	2265
4	631.7799	1.7335e+03	6696
5	737.2993	1.5035e+03	5452
6	764.3227	1.9789e+03	3558
7	852.8252	1.7201e+03	6649
8	887.9353	1.2741e+03	4375
9	929.5465	1.5269e+03	5599

3、计算圆盘中心点坐标

选定第 1,13,28,40 点计算中心点坐标，然后计算交点(中心点坐标)，存于

ldftInfo.centreInfo

实现代码如下：

```
Lx=[leftInfo.coodInfo(1,2) leftInfo.coodInfo(13,2)];
Ly=[leftInfo.coodInfo(1,1) leftInfo.coodInfo(13,1)];
Rx=[leftInfo.coodInfo(28,2) leftInfo.coodInfo(40,2)];
Ry=[leftInfo.coodInfo(28,1) leftInfo.coodInfo(40,1)];
p1=polyfit(Lx,Ly,1);
p2=polyfit(Rx,Ry,1);
Sx=roots(p1-p2);
Sy=polyval(p1,Sx);
leftInfo.centreInfo=[Sy Sx];% 中心点坐标
```

处理结果如下：

leftInfo.centreInfo

	1	2
1	1.9695e+03	1.6521e+03
2		

4、对控制点分组

将选择出来的未分组的控制点与中心点进行一次幂拟合，得到系数，遍历所有控制点，寻找在一条直线上的点，之后重复迭代直至遍历所有控制点;距离中心点的距离从小到大进行排序，求二进制编码组，编号，最后存于 leftInfo.coodInfo 第 4 列

实现代码如下：

```
flag=ones(length(leftInfo.coodInfo),1);% 用于标记是否已经分组
flag=-flag;% -1表示未分组
[C,ia]=unique(flag);% 查找-1第一次出现的位置
coodGroup=-ones(1,4);
for j=1:1:15
    if C(1,1)==-1%未标记完毕
        Lx=[leftInfo.centreInfo(1,2) leftInfo.coodInfo(ia(1,1),2)];
        Ly=[leftInfo.centreInfo(1,1) leftInfo.coodInfo(ia(1,1),1)];
        p=polyfit(Lx,Ly,1);% 将选择出来的未分组的控制点与中心点进行一次幂拟合,得到系数
        i=1:1:length(leftInfo.coodInfo);
        [lay,~]=find(abs(p(1).*leftInfo.coodInfo(i,2)+p(2)...
            -leftInfo.coodInfo(i,1))<100);% 遍历所有控制点,寻找在一条直线上的点
        length_lay=length(lay);
        switch(length_lay)% 对有的不足4个元素的数组进行补充
            case 0
                lay=[-1 -1 -1 -1];
            case 1
                lay=[lay' -1 -1 -1];
            case 2
                lay=[lay' -1 -1];
            case 3
                lay=[lay' -1];
            case 4
                lay=lay';
        end
        coodGroup(j,:)=lay;
    else
        break;% 若所有元素均标定,则退出循环
    end
    flag(lay(1,1:1:length_lay))=1;% 更新控制点标签
    [C,ia]=unique(flag);% 更新未分组点位置
end
```

```
code=zeros(size(coodGroup));
for i=1:1:length(coodGroup)
   lex=coodGroup(i,:);
   if lex(1,4)~=-1
      distance=abs(rightInfo.centreInfo(1,1)-rightInfo.coodInfo(lex,1))'+...
        abs(rightInfo.centreInfo(1,2)-rightInfo.coodInfo(lex,2))';
        for leftM=1:1:4% 排序算法
            for n=leftM:1:4
                if (distance(1,n)<distance(1,leftM))
                    tmp=distance(1,n);
                    distance(1,n)=distance(1,leftM);
                    distance(1,leftM)=tmp;
                    tmpnum=coodGroup(i,n);
                    coodGroup(i,n)=coodGroup(i,leftM);
                    coodGroup(i,leftM)=tmpnum;
                end
            end
        end
        lex=coodGroup(i,:);% 求编码组
        area=rightInfo.coodInfo(lex,3);
        if max(area)-min(area)>2780
            avArea=sum(area,'all')./4;
            x=area>avArea;
            code(i,x)=1;
        elseif max(area)>5000
            code(i,:)=1;
        end
        code(i,:)=4.*(sum(code(i,:).*[8 4 2 1],'all')-1)+[3 2 1 0];
        rightInfo.coodInfo(coodGroup(i,:),4)=code(i,:);
   else
        y=lex(:,:)~=-1;
        code(i,:)=Inf;
        rightInfo.coodInfo(coodGroup(i,y),4)=code(i,y);
   end
end
```

处理结果如下：

leftInfo.coodInfo

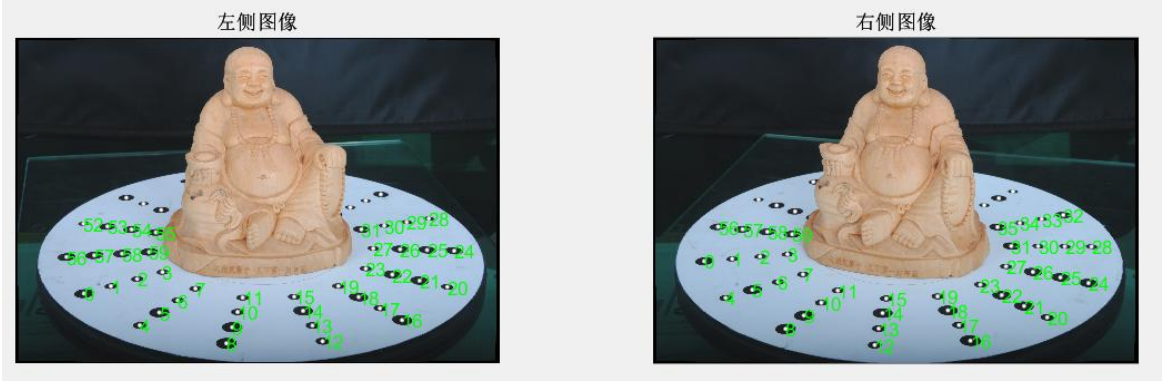
	1	2	3	4
1	409.2966	1.7470e+...	6852	56
2	548.6247	2.0378e+...	8628	0
3	546.3453	1.4805e+...	2265	52
4	631.7799	1.7335e+...	6696	57
5	737.2993	1.5035e+...	5452	53
6	764.3227	1.9789e+...	3558	1
7	852.8252	1.7201e+...	6649	58
8	887.9353	1.2741e+...	4375	Inf
9	929.5465	1.5269e+	5599	54

5、在图像上显示控制点坐标信息并标号

实现代码如下：

```
subplot(1, 2, 1)
imshow(LeftRgb)
title('左侧图像')
hold on;
plot(leftInfo.coodInfo(:,1), leftInfo.coodInfo(:,2), 'w.');
for k = 1:1:length(leftInfo.coodInfo)% 在图上(依绘图顺序)标记
    if leftInfo.coodInfo(k,4)~=Inf
        text(leftInfo.coodInfo(k,1), leftInfo.coodInfo(k,2),num2str(leftInfo.coodInfo(k,4)),'color','g');
    end
end
subplot(1, 2, 2)
imshow(RightRgb)
title('右侧图像')
hold on;
plot(rightInfo.coodInfo(:,1), rightInfo.coodInfo(:,2), 'w.');
for k = 1:1:length(rightInfo.coodInfo)% 在图上(依绘图顺序)标记
    if rightInfo.coodInfo(k,4)~=Inf
        text(rightInfo.coodInfo(k,1), rightInfo.coodInfo(k,2),num2str(rightInfo.coodInfo(k,4)),'color','g');
    end
end
clear k
```

处理结果如下：



6、计算旋转矩阵和误差方程系数矩阵

实现代码如下：

```
% 求解旋转矩阵R
function R=caculateR(parameter)

R=zeros(3,3);
R(1,1)=cos(parameter.Phi)*cos(parameter.Kappa)-sin(parameter.Phi)*sin(parameter.Omega)*sin(parameter.Kappa);
R(1,2)=-cos(parameter.Phi)*sin(parameter.Kappa)-sin(parameter.Phi)*sin(parameter.Omega)*cos(parameter.Kappa);
R(1,3)=-sin(parameter.Phi)*cos(parameter.Omega);
R(2,1)=cos(parameter.Omega)*sin(parameter.Kappa);
R(2,2)=cos(parameter.Omega)*cos(parameter.Kappa);
R(2,3)=-sin(parameter.Omega);
R(3,1)=sin(parameter.Phi)*cos(parameter.Kappa) + cos(parameter.Phi)*sin(parameter.Omega)*sin(parameter.Kappa);
R(3,2)=-sin(parameter.Phi)*sin(parameter.Kappa) + cos(parameter.Phi)*sin(parameter.Omega)*cos(parameter.Kappa);
R(3,3)=cos(parameter.Phi)*cos(parameter.Omega);

end
```

```
% 求解误差方程系数矩阵A
function A=caculateA(innerParameter,outParameter,x,y,R)
deltax=x-innerParameter.x0;
deltay=y-innerParameter.y0;
f=innerParameter.f;

A=zeros(2,6);
Z_=R(1,3)*(x-outParameter.Xs)+R(2,3)*(y-outParameter.Ys)-R(3,3)*outParameter.Zs;
A(1,1)=(1/Z_)*(R(1,1)*innerParameter.f+R(1,3)*deltax);
A(1,2)=(1/Z_)*(R(2,1)*innerParameter.f+R(2,3)*deltax);
A(1,3)=(1/Z_)*(R(3,1)*innerParameter.f+R(3,3)*deltax);
A(2,1)=(1/Z_)*(R(1,2)*innerParameter.f+R(1,3)*deltay);
A(2,2)=(1/Z_)*(R(2,2)*innerParameter.f+R(2,3)*deltay);
A(2,3)=(1/Z_)*(R(3,2)*innerParameter.f+R(3,3)*deltay);

subKappa=deltax*cos(outParameter.Kappa)-deltay*sin(outParameter.Kappa);
sumKappa=deltax*sin(outParameter.Kappa)+deltay*cos(outParameter.Kappa);
A(1,4)=deltay*sin(outParameter.Omega)-(deltax/f*subKappa + f*cos(outParameter.Kappa))*cos(outParameter.Omega);
A(1,5)=-f*sin(outParameter.Kappa)-deltax/f*sumKappa;
A(1,6)=deltay;
A(2,4)=-deltax*sin(outParameter.Omega)-(deltay/f*subKappa - f*sin(outParameter.Kappa))*cos(outParameter.Omega);
A(2,5)=-f*cos(outParameter.Kappa) - deltay/f*sumKappa;
A(2,6)=-deltax;

end
```

7、 计算外部参数

实现代码如下：

```
flag=1;
s=1;% 控制迭代次数以防止死循环
while flag&&s<=50
    LR=caculateR(LOutParameter);% 计算旋转矩阵
    X=point.data(leftInfo.coodInfo(1,4)+1,2);
    Y=point.data(leftInfo.coodInfo(1,4)+1,3);% 控制点世界坐标
    X_=LR(1,1)*(X-LOutParameter.Xs)+LR(2,1)*(Y-LOutParameter.Ys)-LR(3,1)*LOutParameter.Zs;
    Y_=LR(1,2)*(X-LOutParameter.Xs)+LR(2,2)*(Y-LOutParameter.Ys)-LR(3,2)*LOutParameter.Zs;
    Z_=LR(1,3)*(X-LOutParameter.Xs)+LR(2,3)*(Y-LOutParameter.Ys)-LR(3,3)*LOutParameter.Zs;
    lB=caculateA(innerParameter,LOutParameter,leftInfo.coodInfo(1,1),leftInfo.coodInfo(1,2),LR);% 依公式求解B
    ll=[leftInfo.coodInfo(1,1)-innerParameter.x0+(innerParameter.f*(X_./Z_));% 计算l
        innerParameter.height-leftInfo.coodInfo(1,2)-innerParameter.y0+innerParameter.f.*(Y_./Z_)];
    for i=2:1:length(leftInfo.coodInfo)% 基本就是重复上述步骤直到满足条件后退出即可
        if leftInfo.coodInfo(i,4)~=inf
            X=point.data(leftInfo.coodInfo(i,4)+1,2);
            Y=point.data(leftInfo.coodInfo(i,4)+1,3);
            lB=[lB;
                caculateA(innerParameter,LOutParameter,leftInfo.coodInfo(i,1),...
                innerParameter.height-leftInfo.coodInfo(i,2),LR)];
            X_=LR(1,1)*(X-LOutParameter.Xs)+LR(2,1)*(Y-LOutParameter.Ys)-LR(3,1)*LOutParameter.Zs;
            Y_=LR(1,2)*(X-LOutParameter.Xs)+LR(2,2)*(Y-LOutParameter.Ys)-LR(3,2)*LOutParameter.Zs;
            Z_=LR(1,3)*(X-LOutParameter.Xs)+LR(2,3)*(Y-LOutParameter.Ys)-LR(3,3)*LOutParameter.Zs;
            ll=[ll;
                leftInfo.coodInfo(i,1)-(innerParameter.x0-(innerParameter.f.*(X_./Z_)));
                innerParameter.height-leftInfo.coodInfo(i,2)-(innerParameter.y0-(innerParameter.f.*(Y_./Z_)))];
        end
    end
    xl=(lB'*lB)\(lB'*ll);
    flag=(abs(xl(4,1))>=1e-4||abs(xl(5,1))>=1e-4||abs(xl(6,1))>=1e-4);
    s=s+1;
    LOutParameter.Xs=LOutParameter.Xs+xl(1,1);% 更新外方位元素
    LOutParameter.Ys=LOutParameter.Ys+xl(2,1);
    LOutParameter.Zs=LOutParameter.Zs+xl(3,1);
    LOutParameter.Phi=LOutParameter.Phi+xl(4,1);
    LOutParameter.Omega=LOutParameter.Omega+xl(5,1);
    LOutParameter.Kappa=LOutParameter.Kappa+xl(6,1);
end
```

处理结果如下：

LOutParameter

1x1 struct 包含 6 个字段

字段	值
Xs	350.0295
Ys	519.8801
Zs	299.9541
Phi	-0.9172
Omega	-0.8727
Kappa	2.1136

ROutParameter

1x1 struct 包含 6 个字段

字段	值
Xs	130.0102
Ys	610.1385
Zs	300.0758
Phi	-0.4455
Omega	-1.1204
Kappa	2.6605

8、计算控制点的三维坐标和误差

实现代码如下：

```
%计算左右片在地辅坐标系中旋转矩阵的方向余弦
LR=caculateR(LOutInfo);
RR=caculateR(ROutInfo);
%计算基线分量
Bx=ROutInfo.Xs-LOutInfo.Xs;
By=ROutInfo.Ys-LOutInfo.Ys;
Bz=ROutInfo.Zs-LOutInfo.Zs;
for i=1:1:length(point.data)
    %寻找同名点
    l=(leftInfo.coodInfo(:,4)==i-1);
    r=(rightInfo.coodInfo(:,4)==i-1);
    if sum(l,'all')==0||sum(r,'all')==0
       point.data(i,6:1:8)=0;
       continue;
    end

    %计算像点的像空间辅助坐标
    LS=LR*[leftInfo.coodInfo(l,1)-innerInfo.x0;(innerInfo.height-leftInfo.coodInfo(l,2))-innerInfo.y0;-innerInfo.f];
    RS=RR*[rightInfo.coodInfo(r,1)-innerInfo.x0;(innerInfo.height-rightInfo.coodInfo(r,2))-innerInfo.y0;-innerInfo.f];
    %计算投影系数
    LN=(Bx*RS(3,1)-Bz*RS(1,1))/(LS(1,1)*RS(3,1)-LS(3,1)*RS(1,1));
    RN=(Bx*LS(3,1)-Bz*LS(1,1))/(LS(1,1)*RS(3,1)-LS(3,1)*RS(1,1));
    %计算地面点的地面坐标
    point.data(i,6)=LOutInfo.Xs+LN*LS(1,1);
    point.data(i,7)=(LOutInfo.Ys+ROutInfo.Ys+LN*LS(2,1)+RN*RS(2,1))/2;
    point.data(i,8)=LOutInfo.Zs+LN*LS(3,1);
    %计算误差
    point.data(i,9:1:11)=point.data(i,2:1:4)-point.data(i,6:1:8);
end
point.data(61,9:11)=sum(point.data(:,9:1:11));
```

四、相关结果

1、影像上提取控制点并编号

左侧图像

右侧图像

2、计算每张影像的外部参数（后方交会）

LOutParameter

1x1 struct 包含 6 个字段

字段	值
Xs	350.0295
Ys	519.8801
Zs	299.9541
Phi	-0.9172
Omega	-0.8727
Kappa	2.1136

ROutParameter

1x1 struct 包含 6 个字段

字段	值
Xs	130.0102
Ys	610.1385
Zs	300.0758
Phi	-0.4455
Omega	-1.1204
Kappa	2.6605

3、计算控制点的三维坐标（前方交会）

	序号	X	Y	Z		X’	Y’	Z’	X-X’	Y-Y’	Z-Z’
1	0	132.5000	0	0	1	133.5945	-1.3232	-0.3298	-1.0945	1.3232	0.3298
2	1	114	0	0	2	114.9542	-1.1385	-0.2591	-0.9542	1.1385	0.2591
3	2	95.5000	0	0	1	96.3180	-0.9413	-0.1932	-0.8180	0.9413	0.1932
4	3	77	0	0	2	77.7693	-0.7028	-0.1329	-0.7693	0.7028	0.1329
5	4	121.0448	53.8926	0	2	121.9088	52.6191	-0.4320	-0.8640	1.2735	0.4320
6	5	104.1442	46.3680	0	1	104.8623	45.3333	-0.3620	-0.7182	1.0347	0.3620
7	6	87.2436	38.8434	0	2	87.9301	38.0246	-0.2519	-0.6865	0.8188	0.2519
8	7	70.3430	31.3187	0	1	70.8665	30.6874	-0.1760	-0.5235	0.6313	0.1760
9	8	88.6598	98.4667	0	1	89.0662	97.5778	-0.4400	-0.4063	0.8889	0.4400
10	9	76.2809	84.7185	0	2	76.6651	84.1064	-0.3317	-0.3842	0.6121	0.3317
11	10	63.9020	70.9703	0	1	64.2131	70.4710	-0.2305	-0.3111	0.4994	0.2305
12	11	51.5231	57.2222	0	2	51.7947	56.8835	-0.1399	-0.2716	0.3386	0.1399
13	12	40.9447	126.0150	0	2	40.8287	125.7458	-0.2335	0.1160	0.2692	0.2335
14	13	35.2279	108.4204	0	1	35.2063	108.2040	-0.1613	0.0217	0.2164	0.1613
15	14	29.5111	90.8259	0	2	29.4869	90.7499	-0.1274	0.0243	0.0760	0.1274
16	15	23.7943	73.2314	0	1	23.7811	73.2913	-0.0488	0.0132	-0.0599	0.0488
17	16	-13.8500	131.7742	0	1	-14.2257	132.2122	0.0744	0.3756	-0.4380	-0.0744
18	17	-11.9163	113.3755	0	2	-12.2965	113.8211	0.0823	0.3802	-0.4456	-0.0823
19	18	-9.9825	94.9768	0	1	-10.2669	95.4090	0.0768	0.2844	-0.4322	-0.0768
20	19	-8.0487	76.5782	0	2	-8.3215	77.0413	0.1131	0.2728	-0.4631	-0.1131
21	20	-66.2500	114.7484	0	2	-66.7527	115.7509	0.3942	0.5027	-1.0025	-0.3942
22	21	-57.0000	98.7269	0	1	-57.4861	99.6783	0.3484	0.4861	-0.9514	-0.3484
23	22	-47.7500	82.7054	0	2	-48.1538	83.5960	0.3112	0.4038	-0.8906	-0.3112
24	23	-38.5000	66.6840	0	1	-38.8083	67.5050	0.2873	0.3083	-0.8210	-0.2873
25	24	-107.1948	77.8815	0	1	-107.6040	79.3866	0.6611	0.4093	-1.5051	-0.6611
26	25	-92.2279	67.0075	0	2	-92.6054	68.3466	0.5843	0.3775	-1.3391	-0.5843

28	27	-62.2943	45.2595	0	2	-62.6771	46.2937	0.4255	0.3828	-1.0343	-0.4255
29	28	-129.6046	27.5483	0	2	-129.7269	29.3980	0.8591	0.1223	-1.8497	-0.8591
30	29	-111.5088	23.7019	0	1	-111.7197	25.2928	0.7274	0.2108	-1.5908	-0.7274
31	30	-93.4131	19.8556	0	2	-93.7401	21.2137	0.6219	0.3270	-1.3582	-0.6219
32	31	-75.3174	16.0092	0	1	-75.6180	17.1927	0.5154	0.3006	-1.1835	-0.5154
33	32	-129.6046	-27.5483	0	1	0	0	0	0	0	0
34	33	-111.5088	-23.7019	0	2	0	0	0	0	0	0
35	34	-93.4131	-19.8556	0	1	0	0	0	0	0	0
36	35	-75.3174	-16.0092	0	2	0	0	0	0	0	0
37	36	-107.1947	-77.8816	0	2	0	0	0	0	0	0
38	37	-92.2279	-67.0075	0	1	0	0	0	0	0	0
39	38	-77.2611	-56.1335	0	2	0	0	0	0	0	0
40	39	-62.2943	-45.2595	0	1	0	0	0	0	0	0
41	40	-66.2500	-114.7484	0	1	0	0	0	0	0	0
42	41	-57.0000	-98.7269	0	2	0	0	0	0	0	0
43	42	-47.7500	-82.7054	0	1	0	0	0	0	0	0
44	43	-38.5000	-66.6840	0	2	0	0	0	0	0	0
45	44	-13.8500	-131.7742	0	2	0	0	0	0	0	0
46	45	-11.9162	-113.3755	0	1	0	0	0	0	0	0
47	46	-9.9825	-94.9768	0	2	0	0	0	0	0	0
48	47	-8.0487	-76.5782	0	1	0	0	0	0	0	0
49	48	40.9448	-126.0150	0	1	0	0	0	0	0	0
50	49	35.2280	-108.4204	0	2	0	0	0	0	0	0
51	50	29.5111	-90.8259	0	1	0	0	0	0	0	0
52	51	23.7943	-73.2313	0	2	0	0	0	0	0	0
53	52	88.6598	-98.4667	0	2	0	0	0	0	0	0
54	53	76.2809	-84.7185	0	1	0	0	0	0	0	0
55	54	63.9020	-70.9703	0	2	0	0	0	0	0	0
56	55	51.5231	-57.2221	0	1	0	0	0	0	0	0
57	56	121.0448	-53.8926	0	1	122.2079	-54.9836	-0.1609	-1.1631	1.0910	0.1609
58	57	104.1442	-46.3680	0	2	105.1927	-47.3015	-0.1240	-1.0485	0.9335	0.1240
59	58	87.2436	-38.8433	0	1	88.1340	-39.6473	-0.1056	-0.8904	0.8040	0.1056
60	59	70.3430	-31.3187	0	2	71.1938	-32.0104	-0.0610	-0.8507	0.6917	0.0610
61	0	0	0	0	0	0	0	0	-6.0576	-2.2883	-2.2763

四、分析总结

这次大作业对我来说是一个很大的挑战，首先是大一编程学得糊里糊涂、所以我的编程能力并不算优秀，再者大二上我也没有选修过 MATLAB 课程，其次是计算机视觉的原理对我来讲也是很晦涩难懂，最后是大二下的课业繁忙难度提升，只能挤时间完成实习内容。但就是这段逆水行舟逆流而上的实习时间里，我获得了真正的进步。

刚开始接手这个大作业，对最基础的前方交会后方交会原理不是很清楚，花了一周半时间看书、查阅网上资料后终于大致能够理解。紧接着是学习 MATLAB 基本语法，我没有选择系统的看网课学习，而是选择对照网上的计算机视觉代码一行一行抄写、查询每一个不理解的函数、逐个查看运行结果，花了近一个半月从最开始的一头雾水到后来能够掌握 MATLAB 计算机视觉相关语法。对我来说最富有挑战的一步是将课本上前方交会、后方交会的原理转化为 MATLAB 代码，首先必须对原理了熟于心，然后是编写代码得一丝不苟富有耐心，在这样的编写过程之中，不仅巩固了课本上的相关知识，而且锻炼了自己编程能力，恰恰印证了古人的诗句“纸上得来终觉浅，绝知此事要躬行”。

这份实习报告确实算不上尽善尽美，代码也有许多借鉴网上的部分，但是在这个实习过程中学到的技能和知识是真真正正内化成了我自己的东西。很感谢这次实习机会，正所谓有压力才有动力，让我逼了自己一把，逼自己跳脱舒适圈直面自己的薄弱。