摄影测量学

课程实习报告

学生姓名 X

学　　院 李四光学院

班　　级 201226

学　　号 2022100X

授课教师 XX

2024 年 11 月 8 日

目录

1 单像空间后方交会

1.1 实验原理与流程

1.1.1 实验原理

1.1.2 流程图

单像空间后方交会的流程图如下：

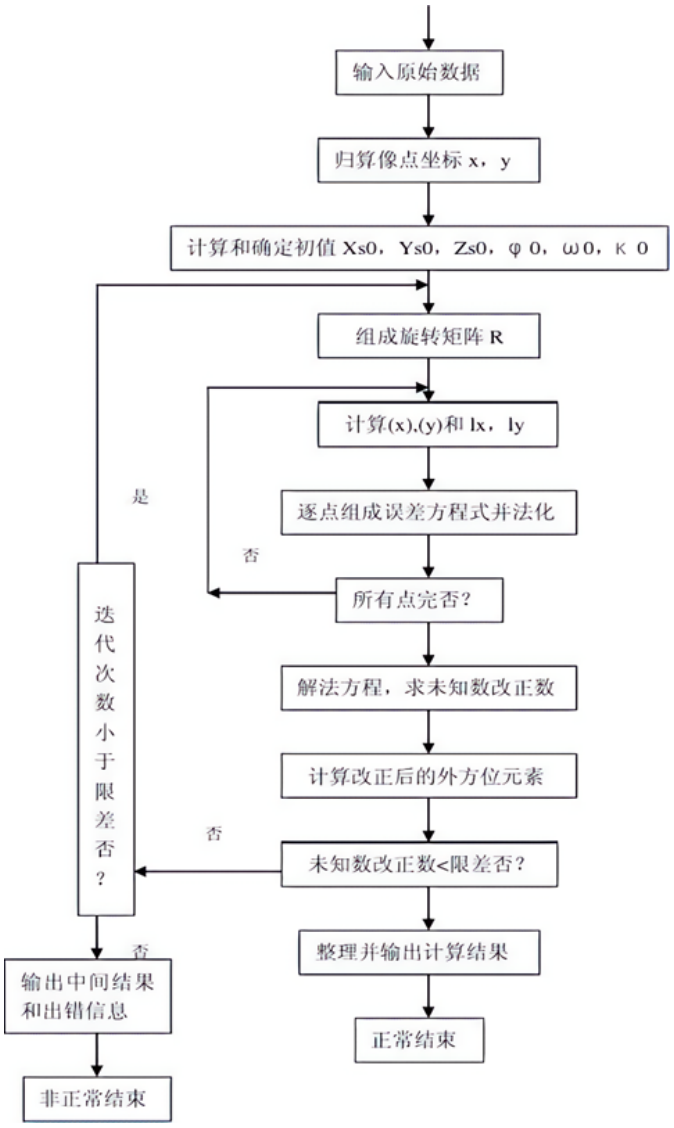

图 1 单像空间后方交会流程图

1.2 实习数据与代码部分

1.2.1 实习数据

摄影机主距 $f = 153.24mm$,$x_0 = 0$,$y_0 = 0$, 像片比例尺为 1:50000，有四对点的像点坐标与相应的地面坐标如下表。

点号	像点坐标		地面坐标		
	x(mm)	y(mm)	X(m)	Y(m)	Z(m)
1	-86.15	-68.99	36589.41	25273.32	2195.17
2	-53.40	82.21	37631.08	31324.51	728.69
3	-14.78	-76.63	39100.97	24934.98	2386.50
4	10.46	64.43	40426.54	30319.81	757.31

图 2 单像空间后方交会数据示意

1.2.2 实习要求

1.2.3 实习代码

1.3 实习结果与数据分析

1.3.1 实习结果

2 写在最后

2.1 发布地址

- Github: https://github.com/HMBlankcat/

附录 A 文件列表

表 1 文件列表

文件名	功能描述
单像空间后方交会.py	单像空间后方交会程序代码
q2.m	问题二程序代码
q3.m	问题三程序代码
q4.m	问题四程序代码

附录 B 代码

单像空间后方交会.py

```python
import numpy as np

#摄影机主距、内方位元素、相片比例尺
fk=153.24
m=50000.0
x0=0.0
y0=0.0
center=np.matrix([x0,y0])

#地面点坐标
mat_ground=np.matrix([[36589.41,25273.32,2195.17],
                      [37631.08,31324.51,728.69],
                      [39100.97,24934.98,2386.50],
                      [40426.54,30319.81,757.31]])
#像点坐标
mat_photo=np.matrix([[-86.15,-68.99],
                     [-53.40,82.21],
                     [-14.78,-76.63],
                     [10.46,64.43]])

#外方位元素
PhotoCenter=np.sum(mat_ground,axis=0)/4 #计算的是地面点的中心

PhotoCenter[0,2]=fk*m/1000.0 #摄影机的高度
H=PhotoCenter[0,2]

```

```
27 print('PhotoCenter:',PhotoCenter)
28
29 #外方位角元素phi omega kapa
30 phi=omega=ka=0 #初始化
31 Angle=np.matrix([phi,omega,ka])
32 #设置改正数组，Xs，Ys，Zs，phi,omega,kapa
33 Iteration=0 #迭代次数
34 while True:
35     Iteration+=1
36     print('这是第',Iteration,'次迭代~')
37     #计算旋转矩阵
38     R=np.matrix([[np.cos(phi)*np.cos(ka)-np.sin(phi)*np.sin(omega)*np.sin(ka
           )     , np.cos(omega)*np.sin(ka)     , np.sin(phi)*np.cos(ka)+np.cos(
           phi)*np.sin(omega)*np.sin(ka)],
39                  [-np.cos(phi)*np.sin(ka)-np.sin(phi)*np.sin(omega)*np.cos(ka
                        )    , np.cos(omega)*np.cos(ka)     , -np.sin(phi)*np.sin(
                        ka)+np.cos(phi)*np.sin(omega)*np.cos(ka)],
40                  [-np.sin(phi)*np.cos(omega)
                                                          , -np.sin(omega)
                                    , np.cos(phi)*np.cos(omega)]])
41
42     a1, b1, c1 = R[0, 0], R[0, 1], R[0, 2]
43     a2, b2, c2 = R[1, 0], R[1, 1], R[1, 2]
44     a3, b3, c3 = R[2, 0], R[2, 1], R[2, 2]
45
46     Ra=np.matrix([[a1,b1,c1],
47                   [a2,b2,c2],
48                   [a3,b3,c3]])
49     print('Ra:', Ra)
50
51     #常数项lx，ly
52     lxy=np.zeros_like(mat_photo)
53     #Zb
54     Zb=np.zeros((mat_ground.shape[0],1))
55     #A系数矩阵
56     A=np.zeros((2*mat_ground.shape[0],6))
57
58     #共线条件方程
59     for i in range(mat_photo.shape[0]):
60         lx=mat_photo[i,0]+fk*(a1 * (mat_ground[i, 0] - PhotoCenter[0, 0]) +
               b1 * (mat_ground[i, 1] - PhotoCenter[0, 1]) + c1 * (
61                          mat_ground[i, 2] - PhotoCenter[0, 2])) / (
62                          a3 * (mat_ground[i, 0] - PhotoCenter[0, 0]) + b3 * (
```

```
                            mat_ground[i, 1] - PhotoCenter[0, 1]) + c3 * (
63                          mat_ground[i, 2] - PhotoCenter[0, 2]))
64          ly=mat_photo[i,1]+fk*(a2 * (mat_ground[i, 0] - PhotoCenter[0, 0]) +
                b2 * (mat_ground[i, 1] - PhotoCenter[0, 1]) + c2 * (
65                      mat_ground[i, 2] - PhotoCenter[0, 2])) / (
66                       a3 * (mat_ground[i, 0] - PhotoCenter[0, 0]) + b3 *
                             (mat_ground[i, 1] - PhotoCenter[0, 1]) + c3 * (
67                           mat_ground[i, 2] - PhotoCenter[0, 2]))
68
69          lxy[i,0]=lx
70          lxy[i,1]=ly
71          x=mat_photo[i,0]
72          y=mat_photo[i,1]
73
74          Zb[i,0]=a3*(mat_ground[i,0]-PhotoCenter[0,0])+b3*(mat_ground[i,1]-
                PhotoCenter[0,1])+c3*(mat_ground[i,2]-PhotoCenter[0,2])
75          A[2 * i, 0] = (a1 * fk + a3 * x) / Zb[i, 0]
76          A[2 * i, 1] = (b1 * fk + b3 * x) / Zb[i, 0]
77          A[2 * i, 2] = (c1 * fk + c3 * x) / Zb[i, 0]
78          A[2 * i, 3] = y * np.sin(omega) - (x * (x * np.cos(ka) - y * np.sin(
                ka)) / fk + fk * np.cos(ka)) * np.cos(omega)
79          A[2 * i, 4] = -fk * np.sin(ka) - x * (x * np.sin(ka) + y * np.cos(ka
                )) / fk
80          A[2 * i, 5] = y
81          A[2 * i + 1, 0] = (a2 * fk + a3 * y) / Zb[i, 0]
82          A[2 * i + 1, 1] = (b2 * fk + b3 * y) / Zb[i, 0]
83          A[2 * i + 1, 2] = (c2 * fk + c3 * y) / Zb[i, 0]
84          A[2 * i + 1, 3] = -x * np.sin(omega) - (y * (x * np.cos(ka) - y * np
                .sin(ka) )/ fk - fk * np.sin(ka)) * np.cos(omega)
85          A[2 * i + 1, 4] = -fk * np.cos(ka) - y * (x * np.sin(ka) + y * np.
                cos(ka)) / fk
86          A[2 * i + 1, 5] = -x
87
88      l=np.matrix(lxy).reshape(8,1) #求转置
89      print('Zb:',Zb)
90      print('A:',A)
91      print('l:',l)
92      A=np.matrix(A)#将A转换为矩阵
93      #最小二乘法求解
94      mat1=((A.T*A).I*A.T*l)
95      delta=mat1
96      print('delta:',delta)
97      dphi, domega, dka = delta[3, 0], delta[4, 0], delta[5, 0]
```

```
98      phi += dphi
99      omega += domega
100     ka += dka
101     PhotoCenter[0, 0] += delta[0, 0]
102     PhotoCenter[0, 1] += delta[1, 0]
103     PhotoCenter[0, 2] += delta[2, 0]
104     H=PhotoCenter[0,2]
105     print('PhotoCenter:',PhotoCenter)
106     print('phi:',phi)
107     print('omega:',omega)
108     print('ka:',ka)
109     #迭代停止条件
110     if np.abs(delta).max()< 3e-5:
111         print('delta:', delta)
112         print('dphi', dphi)
113         print('domega', domega)
114         print('dka', dka)
115         break
116
117 print('phi:',phi)
118 print('omega:',omega)
119 print('ka:',ka)
120 print('PhotoCenter:',PhotoCenter)
121 # 计算残差向量v
122 v = l - A * delta
123 #残差的中误差计算
124 n = 2 * mat_ground.shape[0]  #观测方程的数量
125 u = 6  #未知数的数量
126 sigma = np.sqrt((v.T * v)[0, 0] / (n - u))  #中误差计算
127
128 print('残差向量v:',v)
129 print('中误差sigma:',sigma)
```