University of Cape Town

School of Architecture and Planning and Geomatics

Geomatics Division

APG4005F
Engineering Surveying
Assignment 1

BSc Geomatics

By: Jayren Kadamen
KDMJUN001

Course Conveynor:
George Sithole

March 13, 2015

Abstract

An architect wishes to determine if the ceiling to an ancient building forms a hemisphere. This report discusses the use of photogrammetry, least squares, and statistical analysis in order to solve this problem.

This assignment is a powerful example of how different disciplines within the Geomatics umbrella come together in order to solve a problem from another discipline.

Contents

1 Introduction

Geomatics is a multidisciplinary profession. Problems within the Geomatics field, much like problems approached in all Engineering disciplines, require a systematic approach in order to reach a solution. Once that solution has been reached it needs to be assessed as to whether the conclusion drawn is indeed correct.

For this assignment an architect desired to know whether the ceiling to an ancient building formed a perfect hemisphere. In order to solve this problem photogrammetric methods were used to obtain points on the dome, a least squares approach was used to determine the adjusted values for the parameters that define a dome (origin and radius), and lastly statistical testing was used to assess whether we had drawn the correct conclusion concerning the shape of the dome.

2 Background to Problem

An architect wishes to know whether a ceiling to an ancient building is indeed a perfect hemisphere as its reputation suggests. The points were measured to a precision of +/- 3mm.

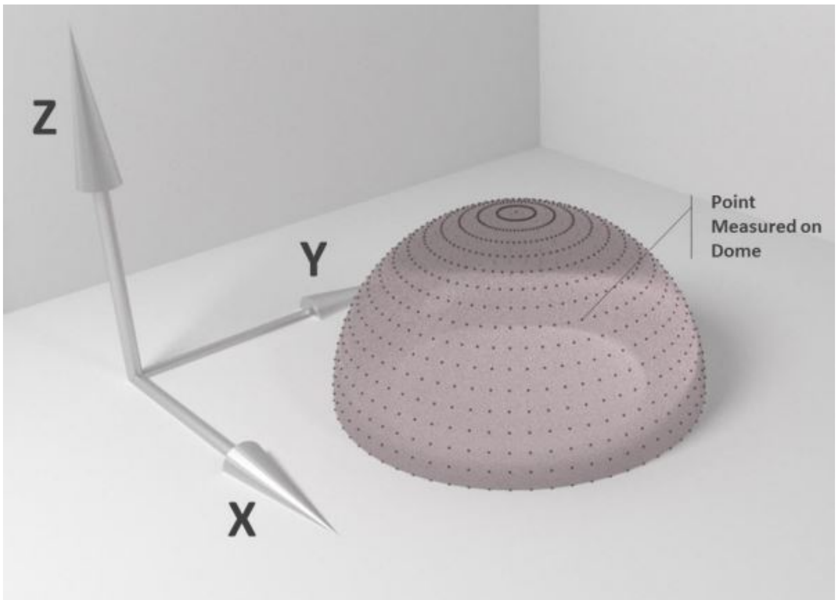

Figure 1: Points measured on the ceiling surface"

3 Problem Statement

To ascertain if the dome forms a hemisphere using points on the dome that were measured to a precision of +/- 3mm.

4 Methodology

The assignment was broken up into three major components which will be detailed below in the upcoming subsections.

The following software suites/packages were used:

- Excel to view the .csv file
- Python 2.7.3 with Enthought Python Distribution
  - Numpy 1.6.1
  - Scipy 0.10.1
  - Matplotlib 1.1.0
- TeXstudio using Qt Version 4.8.5 for this report

4.1 Calculating Provisional Values

To define a hemisphere one requires an origin and a radius. These points are considered our provisional values; (x_0, y_0, z_0, r). In order to calculate them the data file was read in, arrays were populated with respect to their parameter (e.g. x_{array} for x values) and the minimum and maximum values of the arrays were recorded. The difference between the minimum and maximum values for x,y, and z were used to calculate the provisional values for the origin and radius:

```
x0 = (maxx-minx)/2 + minx
y0 = (maxy-miny)/2 + miny
z0 = minz
r0 = ((maxx-minx)/2+(maxy-miny)/2+(maxz-minz))/3
```

4.2 Plotting

The lists of observations were then used to plot the measured points to save one from having to open the data file and loop or cycle through it again. Plotting was achieved using the 3D projections from the *matplotlib* library. The colour map was defined in relation to the z-values.

The plot is only output once the last iteration has completed and the statistical computation completed so that the program can execute fully without waiting for the plot to close before continuing.

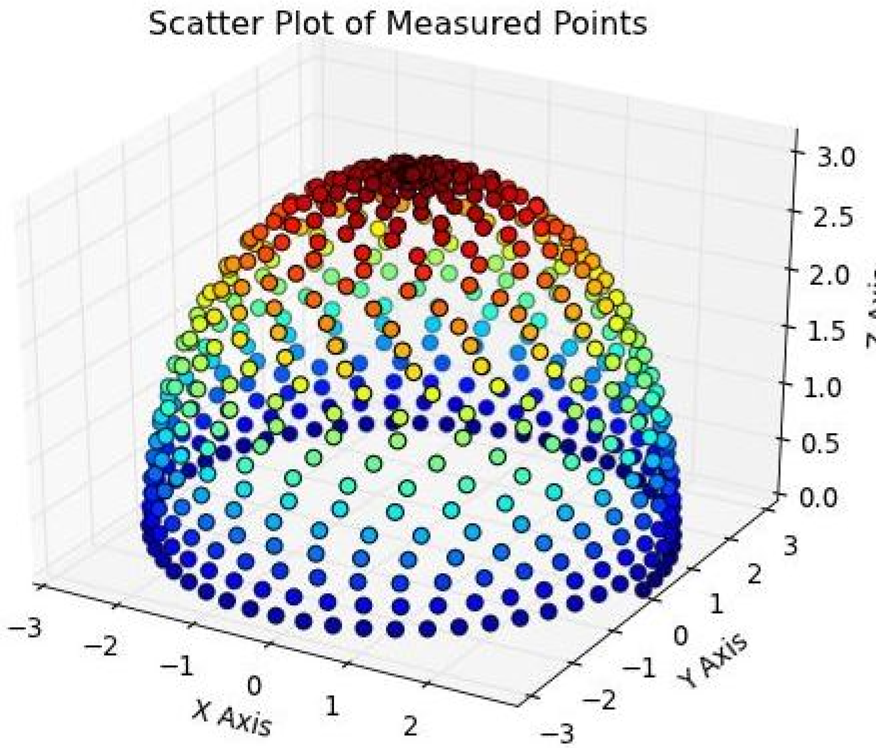

Figure 2: Program output of Measured Points

4.3 Populating Matrices

The provisional coordinates (x_0, y_0, z_0, r) were sent to another class designed solely to perform the least squares calculation and the statistical analysis. It is structured such that the provisional values are set as constructors. The equation for a hemisphere is as follows:

$$f = (((x - x_0)^2) + ((y - y_0)^2) + ((z - z_0)^2)) - r^2 \tag{1}$$

4.3.1 Populating A Matrix

Populating the A matrix (coefficients of the unknowns) required no special calculation or preparation. The A matrix consists of the coefficients of the unknowns:

$$\begin{pmatrix} \frac{\partial f^1}{\partial x_0} & \frac{\partial f^1}{\partial y_0} & \frac{\partial f^1}{\partial z_0} & \frac{\partial f^1}{\partial r} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f^n}{\partial x_0} & \frac{\partial f^n}{\partial y_0} & \frac{\partial f^n}{\partial z_0} & \frac{\partial f^n}{\partial r} \end{pmatrix}$$

Where f^1 is the function for the first observation and f^n is the function for observation n.

4.3.2 Populating B Matrix

Populating the B matrix (coefficients of the residuals) was achieved by using the quasi-parametric model. This follows the typical format of the general adjustment model:

$$Ax + Bv + w = 0 \tag{3}$$

The primary difference is the use of the quasi-weight matrix $\overline{P}$:

$$\overline{P} = (BP^{-1}B^T)^{-1} \tag{4}$$

This quasi-weight can only be used when each observation only appears once in each condition equation which is true in this case. By using the quasi weight we substantially reduce the computational resources required to perform the adjustment. This is achieved by only populating the diagonal of matrix B:

$$\overline{P} = (BP^{-1}B^T)^{-1} = \begin{pmatrix} \frac{1}{\sum[\frac{(aa)}{p}]} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{\sum[\frac{(bb)}{p}]} & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & \frac{1}{\sum[\frac{(nn)}{p}]} \end{pmatrix}$$

where

$$\frac{1}{\sum[\frac{(aa)}{p}]} = \frac{1}{\frac{(x_a - x_0)^2}{p_{x_a}} + \frac{(y_a - y_0)^2}{p_{y_a}} + \frac{(z_a - z_0)^2}{p_{z_a}}}$$

The use of aa denotes the first observation whilst nn denotes the last observation.

Once matrix $\overline{P}$ has been populated the solution vector is given by:

$$x = -(A^T\overline{P}A)^{-1}A^T\overline{P}w \tag{7}$$

For this problem the weight matrix was set to identity so all p values above are equal to one.

4.3.3 Populating w Vector

The w vector was populated as per normal for the general case of the least squares adjustment using:

$$w = (((x - x_0)^2) + ((y - y_0)^2) + ((z - z_0)^2)) - r^2 \tag{10}$$

4.4 Solving the x Matrix

In the program code obtaining the vector of discrepancies, x was obtained by doing the following:

```
#Usually B*P^-1*B.T but weight matrix is set to identity matrix
#Calculation of M is required before M can be inversed
  M = B*(B.T)

#Calculation of the vector of discrepancies
  X = -(A.T*(M.I)*A).I*A.T*M.I*W
```

The adjustments were added to the provisional values and the process was iterated until the magnitude of the adjustments were less than 0.005. For this data set only two iterations were necessary until adjustment values became insignificant (fell below magnitude 0.005).

The vector of correlates, k and the vector of residuals, v were calculated using:

$$k = -(BP^{-1}B^T)^{-1}(Ax + w) \tag{8}$$

$$v = P^{-1}B^T k \tag{9}$$

Recall that $P = I$ in all the above equations.

4.5 Calculation of Variance Matrices and Global Test

Once the final iteration is complete the global check and the multiple variance matrices have to be calculated. The covariance matrix of the unknowns are used in the calculation of the standard deviation of the adjusted radius which is later used in the hypothesis test. The code for these matrices can be found below:

```
  #Cofactor Matrix
  Qxx = ((A.T)*(M.I)*A).I

  S = (M.I)*A*Qxx
   #Cofactor of the correlates
  Qkk = M.I - (M.I)*A*(S.T)

  #Reference Variance
  Var = ((V.T)*V)/(B_Col-3)

  #Covariance matrix of discrepancies w
  Eww = (Var.item((0,0)))*B*B.T

  #Covariance of Unknowns
  Exx = (Var.item((0,0)))*Qxx
#Standard Deviation of the Radius
  SigmaR = math.sqrt(Exx.item((0,0)))
```

4.6 Hypothesis Test

The hypothesis test was calculated using the scipy.stats library in python. Our null hypothesis stated that the variance of our population is equal to our sample variance. This was then tested at a significance level that the user was prompted to provide. Below is the function created to calculate the Chi-Squared test for this dataset:

```
def stats(self,SigmaR,B_Col):
    #Chi Squared test
    print('Our Null Hypothesis states that the variance of our population = sample
        variance')

    #Variance of our radius from measured points
    observed = SigmaR

    #Expected variance (3mm per x,y,z observation)
    expected = .003**2+.003**2+.003**2

  #Calculation of degrees of freedom
    dof = B_Col-1

    #Calculation of test statistics
    teststatx = B_Col*((observed - expected)**2/expected)
    teststatx1 = dof*(observed/expected)

    #User is prompted to input desired significance level
    significance = np.float(input('Please specify the significance level: '))

    print(teststatx),
    print(teststatx1)

    #Using built in scipy.stats.chi2 function instead of looking up values on a
        table
    mean, var, skew, kurt = chi2.stats(dof, moments='mvsk')
    Chi = chi2.ppf((1-significance),dof)

  #If our sampled variance is greater than the population variance at the chosen
      significance level then we reject the null hypothesis at that significance
      level
    if teststatx > Chi:
        print 'We reject the null hypothesis at the ',significance,'significance
            level'

    else:
        print 'We fail to reject the null hypothesis at the
            ',significance,'significance level'

    print(teststatx, dof)
```

5 Results

The program only needed to iterate twice before obtaining the desired precision of magnitude 0.005. The following results for the adjusted unknowns x_0, y_0, z_0, r are as follows:

- $x_0 = -0.000$
- $y_0 = 0.002$
- $z_0 = 0.032$
- $r = 3.084$

The points above (to the exclusion of r) define the origin of the hemisphere and have been rounded down to the nearest millimetre. This is due to the fact that the population variance was given to $+/-3mm$

Our reference variance of 0.00076531 illustrates that there is significant redundancy concerning the number of points measured in order to perform the adjustment and distribute random errors across many more points than needed which will provide a better result.

At a significance level of 5% we fail to reject the null hypothesis. This means that the sample radius does not differ significantly from our population variance of 3mm.