MANUAL OF ODIS FITTING FUNCTIONS [1]

Contents

Contents 1

Introduction 2
Problem description 2
Implement tool 2
Note 2

Matlab Code Comments 3

C++ Code Comments 5

Appendix 12
Data for Circle Fit 12
Simulation Data for Ellipse Fit 13
Singular Value Decomposition And Fitting 14
Circle Fitting with Known Radius by Iteration 15

Index 16

[1] This document is wrote by Zhen Song, CSOIS, ECE Dept. Utah State Univ. This document is still under construction.

Introduction

Problem description

The following fitting functions are designed to fit laser range data to circle or ellipse. See Figure 1 for the example of circle fit. Dr. Yangquan Chen[2] proposed these algorithms, and wrote the initial Matlab functions[3] to solve the problems.

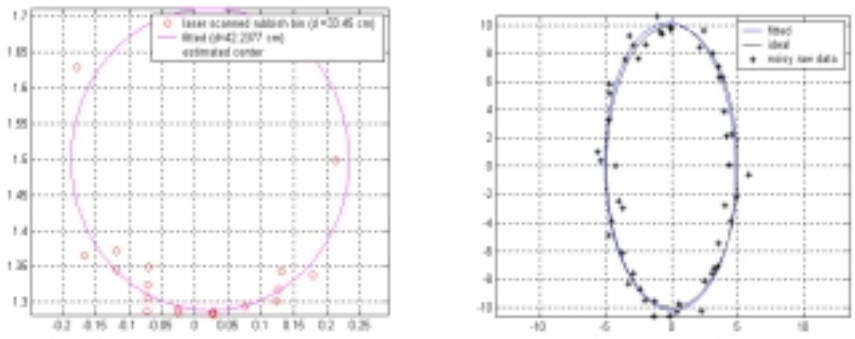
Implementation

Thanks to Robert Davies[4], free C++ matrix library, which is called NewMat, is available on the Internet[5]. The latest version is 10, here I use version 9. For details, please see the PDF or HTML version of the manual. According to the manual, the NewMat support Gun C++ and Microsoft Visual C++, etc.

Note

- The Circle Fit/Ellipse Fit algorithms do not require strict sequence of samplings, and data of an arc can still be fit to a whole circle. See figure 1.
- In this document the X and Y-axis are the same as that of ODIS. See figure 2.
- The fitting functions are included in the file "Fits.cpp" and perform as a part of the library. Thus several lines on Newmat code need to be changed. The files under the directory "CppCode" are modified. The comparisons on those files are in the "Rc" directory , named by the pattern "filename_Diff.htm". A version control tool, CSDiff[6], generates those comparison files.

[2] PPT file is ChenCircle_fit_ODIS2
[3] Available in the same zip file as this document. In directory "MatlabCode"
[4] http://webnz.com/robert/index.html robertd@netlink.co.nz
[5] http://webnz.com/robert/download.html
[6] CSDiff can be downloaded from http://www.componentsoftware.com/csdiff/ for free.



A B

Fig 1: Circle/Ellipse Fit (A) Circle Fit (B) Ellipse Fit

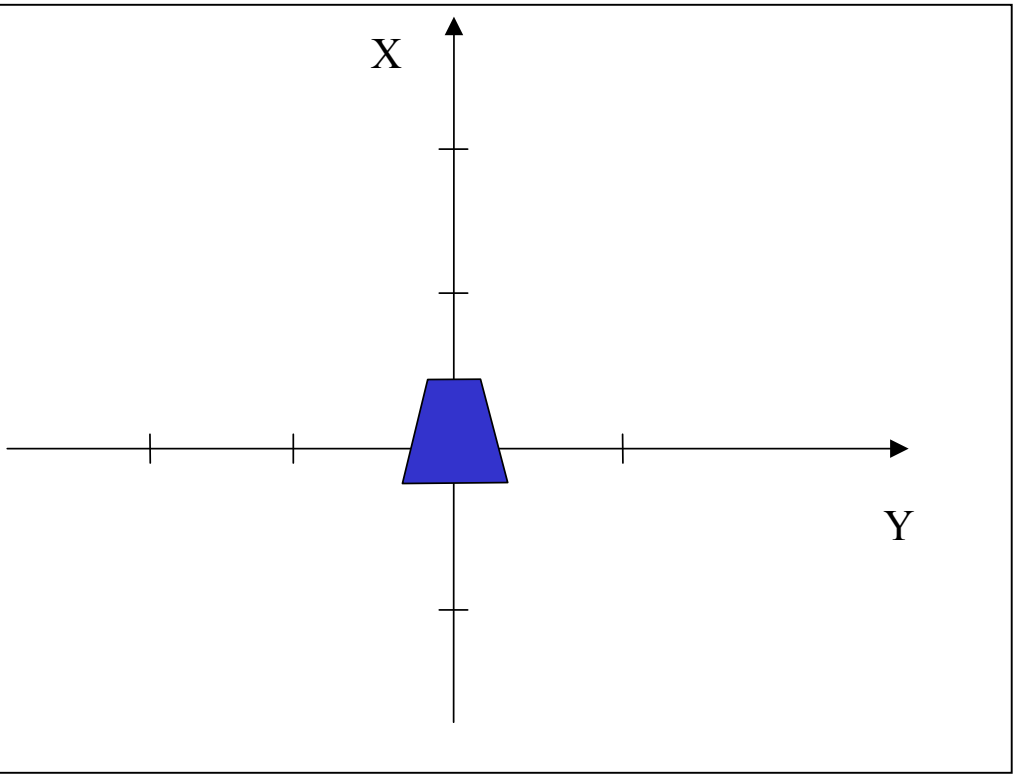


Fig 2: Coordination of ODIS

Matlab Code Comments

1. algcircle

Algebraic circle fit.

- Syntax
```
[z,r] = algcircle(x)
```
- Description
  `x` are points to fit, with the format [y1, x1; y2, x2; …];
  `z` is the origin of the circle, with the format [y;x];
  `r` is the radius of the circle.
- Example
```
a = [
    1.1161    0.1735
    1.0357    0.1528
    1.0357    0.1528
    1.0105    0.1464];
xin=[a(:,2),a(:,1)];
[z, r] = algcircle(xin);
```
- See Also
  AlgCircle, krcircle,

2. krcircle

Circle fit for known radius.

- Syntax
```
z = krcircle(xin,z,rstar);
```
- Description
  `xin` are points to fit, with the format [y1, x1; y2, x2; …];
  `z` is the initial origin of the cirlce, with the format [y;x];
  `rstar` is the actual radius of the circle.
- Example
```
a = [
    1.1161    0.1735
    1.0357    0.1528
    1.0357    0.1528
xin=[a(:,2),a(:,1)];
[z, r] = algcircle(xin);
rstar=.3345/2;
z = krcircle(xin,z,rstar);
```
- See Also
  algcircle, AlgCircle

3. fitellipa

Fit ellipse in the sense of lease square.

- Syntax
```
a=fitellipa(x,y)
```
- Description

`x` and `y` are points to fit;

`a` are the parameters of ellipse: $a(1)x^2 + a(2)xy + a(3)y^2 + a(4)x + a(5)y + a(6) = 0$

- Example

```
X=[ -0.08600004206036
  -0.07794299456979
   2.43242519802460
   2.09758733070173
   3.13534679991087
   3.56316973735175   ];
y=[ 9.84922535631035
   9.64665520238136
   9.58894219243244
   8.40577225886695
   8.00036929745479
   7.02883511552260 ];
a=fitellipa(x,y);
```

- See Also
  FitEllipse, SolveEllipse

C++ Code Comments[7]

4. AlgCircle

Algebraic circle fit.

- Syntax
  [in Fits.cpp]

```
#include "Fits.h"
int AlgCircle(const Matrix & X, circle & Cir)
```

- Description

X is a reference to a N by 2 `Matrix`. First column must be x, second column is y (note: different from algcircle)
Cir is `circle` type reference. This is the circle already fitted. It has three representations. Any of the three is valid. The `circle` is defined in Fits.h as:

```
typedef struct Circle{
   double x0, y0; //(x-x0)^2+(y-y0)^2=r^2
   double Cx, Cy, C; //x^2+Cx x+ y^2+ Cy y+ C=0
   double rho, theta; //polar cordination
   double r;
   double err; //fitting error
}circle;
```

Return: If got correct answer, return 0; If caught error in NewMat library, return –1; If the input parameter is invalid, return –2.
Note: Unlike C syntax, the index of `Matrix` starts from 1.

- Example

```
Real a[]={
1.1161 ,   0.1735,
 1.0357 ,   0.1528,
 1.0357 ,   0.1528,
 1.0105 ,   0.1464,
```

[7] I made small modification to NewMat. The comparison of the different version can be seen under RC directory, in files: include_h_Diff.htm, Newmat_h_Diff.htm, Newmat6_cpp_Diff.htm.

```
    1.0105 ,   0.1464,
    0.9972 ,   0.1050,
    0.9972 ,   0.1050,
    0.9854 ,   0.1025};
Matrix X(8,2);
X<<a;
circle Cir;
if(int ret=AlgCircle(X, Cir)==-1){
   cout<<"lib error"<<endl;
}
else if(ret==-2){
   cout<<"invalid parameter"<<endl;
}
```

- See Also
  algcircle, Fit.h, Fit.cpp, CPPfit.cpp

- Testing

1. Verified by Matlab program algcircle. Used the real laser data set A and B got from Lili Ma (Appendix, page 12), and the output was:

Matlab algcircle

```
z =
   0.01568653329927
   1.13826501277023
r =
   0.17036687906663
err =
   0.19211192914545
z =
   0.02302702160016
   1.49968286447904
r =
   0.21118860540022
err=
   0.34804198928622
```

C++ AlgCircle

```
for case A
(1.138265012770230,0.015686533299272)
r=0.170366879066633
err=0.192111929145481

for case B
(1.499682864479015,0.023027021600139)
r=0.211188605400194
err=0.348041989286223
```

Those data are include in CPPfit.cpp

2.Exception
It is designed to be robust. If the column number of matrix X is not 2, return –1; If fitting points are no more than 2, return –1, since we can not use 2 points to fit circle.
For more exception case, need for test.

- PDL

```
Input matrix X, which has data; and cir, which would return
parameters of circle.
If X does not has exactly 2 columns, or the input data is not
enough,
   return -2, which means parameters are invalid.
End if
```

The first column of X is x, the second one is y.
Question is: find solution of a1(x^2+y^2)+a2 x+ a3 y+ a4=0, by means of least square.
Let matrix B=[x^2+y^2, x, y, 1];

Do singular value decomposition for matrix B, thus B=U D V.t( ).
t( ) means transpose.
Find the minimum element D(m,m) on the diagonal of D.
The column m of V is the solution. i.e., a1=V(1,m), a2=V(2,m), a3=V(3,m), a4=V(4,m).[8]
Transfer between different form: polar form and standard form
During the transformation, take care of the theta (in polar form). Make sure it would not comes to infinity. It is defined as: *x0, y0 are the origin*

1. If x0 and y0 are both 0, theta=0;
2. If x0=0 and y0>0, theta=pi/2;
3. If x0=0 and y0<0, theta=-pi/2;
4. Otherwise theta=atan2(y0,x0).

Then compute error.
Initialize error as 0.
In for loop: Try every elements of the circle
   error=error+abs((x-x0)^2 + (y-y0)^2 -r)
End for loop
Save solution to cir.
Return.

5. KrCircle

Circle fit for known radius.

- Syntax

```
#include "Fits.h"
int KrCircle(const Matrix & x, circle & cir);
```

- Description

`x` is a reference to a N by 2 `Matrix`. First column is x, second column is y.
The definition of `cir` is the same of `AlgCircle`.

Return: If get correct answer, return 0; If caught error in NewMat library, return –1; If input parameter is invalid, return –2.

Note: Unlike Matlab function `krcircle`, the C++ function KrCircle already include AlgCircle, so the coordination of initial origin is not required.

- Example

```
Real a[]={
    1.1161 ,   0.1735,
    1.0357 ,   0.1528,
    1.0357 ,   0.1528,
    1.0105 ,   0.1464,
    1.0105 ,   0.1464,
    0.9972 ,   0.1050,
    0.9972 ,   0.1050,
    0.9854 ,   0.1025};
Matrix X(8,2);
X<<a;
circle Cir;
if(int ret=KirCircle(X, Cir)==-1){
   cout<<"lib error"<<endl;
}
else if(ret==-2){
   cout<<"invalid parameter"<<endl;
```

[8] See appendix for more details.

```
}
```

- See Also
  algcircle, krcircle

- Testing

1. Used the same data for `AlgCircle` testing, and radius is 0.3345/2;

Matlab krcircle

```
z =
   0.01563411608133
   1.13510318561113
rstar =
   0.16725000000000
```

C++ KrCircle

```
(0.015634116081327,1.135103185611130)
assume r=0.167250000000000
error=54.533942128808796
```

- PDL

```
Input matrix X, which has data; and cir, which has radius already
set, and would return parameters of circle.
If X does not has exactly 2 columns, or the input data is not
enough,
   return -2, which means parameters are invalid.
End if
Backup the input radius as true radius.
Do AlgCircle first.
If the error is too large, return -2. This indicates input data are
invalid.
Initial U as the estimated origin.
While norm of the iterator is not small enough [9]
   fake is the estimation error.
   J is the gradient.
   Find solution of: J h+f=0. Since h is unknown, matrix left divide
is applied.
   U is updated by h.
End while loop.
Transfer to different form. As in AlgCircle.
Compute error. As in AlgCircle.
```

6. **FitEllipse**
   Fit ellipse.

- Syntax

```
#include "Fits.h"
int FitEllipse(Matrix & X, ellipse & ell);
```

- Description
  The ellipse is defined as:

```
typedef struct Ellipse{
   double Cx2, Cxy, Cy2, Cx, Cy, C;
   //Cx2 * x^2 + Cxy * x*y + Cy2 * y^2 + Cx * x + Cy * y +C =0
   double x0, y0, rx,ry,theta;
   // \over{(x-x_0)^2}{r_1} + \over{ (y-y_0)^2}{r_2} = 0,
   // and rotate \theta degree.
   double err;//fitting error
}ellipse;
```

[9] See appendix on page15 for details on algorithms.

Return: If get correct answer, return 0; If caught error in NewMat library, return –1; If input parameter is invalid, return –2.

Note: 1. Inside `FitEllipse`, `SolveEllipse` is called, in order to transfer ellipse from polynomial form to standard form.

2. The error in struct `ellipse` is algebra error, which is defined as

$$\sum (a(1)x^2 + a(2)xy + a(3)y^2 + a(4)x + a(5)y + a(6))^2$$

Since it has no geometry meaning, it is more difficult the give a threshold than `AlgCircle` case, where the error is geometry error, which has a real geometry meaning. In the following test data set, the algebra error is about 6.2 for 50 points.

- Example

```
Real ellip[ ]={
-0.08600004206036,   9.84922535631035,
-0.07794299456979,   9.64665520238136,
 2.43242519802460,   9.58894219243244,
 2.09758733070173,   8.40577225886695,
 3.13534679991087,   8.00036929745479, };
 int Row=sizeof(ellip)/sizeof(Real)/2;
 Matrix El(Row,2);
 El<<ellip;
 ellipse el;
 FitEllipse(El, el);
```

- See Also

  fitellipa, SolveEllipse

- Testing

1. Verified by randomly generated data from Matlab program, chendemo. Since the program makes new set of data every time, the data in appendix are used as standard and the following results are based on those data.

Matlab fitellipa (see `fitellipa` *for the definition of* `a`*)*

```
a =
  -0.04000092644031
  -0.00127336790306
  -0.00964837674969
  -0.00975836062049
  -0.00062707201838
   0.99910439615832
>>solveellipse(a)
   rx=10.18613727180503
   ry=4.99837136339120
   x0=-0.12158715466949
   yo=-0.02447286473420
   theta=-1.54983232783560
```

C++ FitEllipse

```
parameters
-0.040000926440313
-0.001273367903062
-0.009648376749694
-0.009758360620494
-0.000627072018379
0.999104396158319
error=6.289606737088382
origin (-0.121587154669531, -0.024472864734140),
rx=10.186137271803190
ry=4.998371363391024
theta=-1.549832327835561
```

- PDL

```
Input matrix Ellipse, and struct el,which would return parameters of
ellipse.
```

```
If X does not has exactly 2 columns, or the input data is not
enough,
   return -2, which means parameters are invalid.
End if
The first column of X is x, the second one is y.
Question is: find solution of a1 x^2+a2 xy +a3 y^2+ a4 x+ a5 y +
a6=0, by means of least square.
Let matrix B=[x^2, xy, y^2, x, y, 1];
Do singular value decomposition for matrix B, thus B=U D V.t( ).
 t( ) means transpose.
Find the minimum element D(m,m) on the diagonal of D.
The column m of V is the solution.
Call SolveEllipse to transfer between different forms.
Then compute error.
Initialize error as 0.
In for loop: Try every elements of the circle
   error=error+abs(a1 x^2+a2 xy +a3 y^2+ a4 x+ a5 y + a6)
End for loop
Save solution to el;
```

7. SolveEllipse

Trans ellipse from polynomial form :

$$a(1)x^2 + a(2)xy + a(3)y^2 + a(4)x + a(5)y + a(6) = 0$$

to standard form:

$$\frac{(x-x_0)^2}{R_x} + \frac{(y-y_0)^2}{R_y} = 0$$, Then rotate an angle of θ

- Syntax

```
#include "Fits.h"
SolveEllipse(ellipse &el);
```

- Description

`el` is a structure to store parameters of an ellipse. For details, see `FitEllipse`.
Return void.

- Example

In FitEllipse( )

```
            Ellipse el;
```

Compute V

```
            el.Cx2=V(1,nMinInd);
            el.Cxy=V(2,nMinInd);
            el.Cy2=V(3,nMinInd);
            el.Cx= V(4,nMinInd);
            el.Cy= V(5,nMinInd);
            el.C = V(6,nMinInd);

            SolveEllipse(el);
```

- Testing

See FitEllipse

8. mldivide

Do matrix left divide.

- Syntax

```
#include "Fits.h"
int mldivide(Matrix & ans, Matrix nominator, Matrix denominator);
```

- Description
  `ans` is the solution of `nominator\denominator.`
  This function uses QR decomposition of NewMat to find solution:

```
Matrix Lf=nominator;
Matrix Rt=denominator;
QRZ(Lf, U);
QRZ(Lf, Rt, M);
Ans=U.i( )*M;
```

- Example

```
Real lf[]={1,2,3,4}, rt[]={ 5,6};
Matrix Lf(2,2),Rt(2,1),Ans;
mldivide(Ans,Lf,Rt);
cout<<Ans;
```

- See Also

9. BaseMatrix::Rank

Find rank of a matrix by SVD approach.

- Syntax

```
#include "Newmat.h"
int BaseMatrix::Rank( )
```

- Description
  Find the rank of a matrix. This function use SVD[10] to find the rank. Matrices are not subject to be square.
  There is a limitation: This function check the diagonal of D matrix of SVD. If the absolute value of any element is too small, (defined by `SMALLEST_POSITIVE` in `include.h`.) it is believe as 0. This approach is valid to normal engineering application, but not valid for pathology system.

- Example

```
Real mat[]={1, 2, 3, 4};
Matrix Mat(2,2);
Mat<<mat;
cout<<Mat.Rank( );
```

- See Also

[10] See appendix at page14

Appendix

Data for Circle Fit

In Matlab format, the real laser data of circle fitting are ([x1 y1; x2 y2])
Data set A:

```
xy = [    1.1161    0.1735
   1.0357    0.1528
   1.0357    0.1528
   1.0105    0.1464
   1.0105    0.1464
   0.9972    0.1050
   0.9972    0.1050
   0.9854    0.1025
   0.9854    0.1025
   0.9854    0.1025
   0.9883    0.0661
   0.9883    0.0661
   0.9745    0.0636
   0.9745    0.0636
   0.9705    0.0629
   0.9719    0.0243
   0.9700    0.0240
   0.9700    0.0240
   0.9700    0.0240
   0.9700    0.0240
   0.9710   -0.0109
   0.9710   -0.0109
   0.9710   -0.0109
   0.9710   -0.0109
   0.9710   -0.0109
   0.9756   -0.0455
   0.9756   -0.0455
   0.9816   -0.0451
   0.9816   -0.0451
   0.9896   -0.0446
   0.9872   -0.0797
   0.9982   -0.0794
   0.9982   -0.0794
   0.9982   -0.0794
   0.9982   -0.0794
   0.9946   -0.1144
   1.0286   -0.1145
   1.0286   -0.1145
   1.0456   -0.1145
   1.0456   -0.1145
   1.1797   -0.1572];
```

Data set B:

```
    xy=[  1.4988,    0.2138
   1.3375,    0.1787
   1.3427,    0.1314
   1.3181,    0.1269
```

1.3181, 0.1269
1.3023, 0.1241
1.3023, 0.1241
1.2937, 0.0757
1.2937, 0.0757
1.2937, 0.0757
1.2937, 0.0757
1.2937, 0.0757
1.2843, 0.0274
1.2843, 0.0274
1.2823, 0.0272
1.2823, 0.0272
1.2863, 0.0276
1.2860, -0.0249
1.2900, -0.0247
1.2900, -0.0247
1.2900, -0.0247
1.2900, -0.0247
1.2880, -0.0724
1.3070, -0.0719
1.3070, -0.0719
1.3240, -0.0713
1.3240, -0.0713
1.3490 , -0.0706
1.3453 , -0.1193
1.3713, -0.1194
1.3713, -0.1194
1.3713, -0.1194
1.3713, -0.1194
1.3661, -0.1680
1.6289, -0.1788]

Simulation Data for Ellipse Fit

This data are randomly generated by Dr. Yangquan Chen's Matlab program. The C++ code is proved correct with these numbers ([x1 y1; x2 y2;…]) :

xy=[-0.08600004206036 9.84922535631035
-0.07794299456979 9.64665520238136
2.43242519802460 9.58894219243244
2.09758733070173 8.40577225886695
3.13534679991087 8.00036929745479
3.56316973735175 7.02883511552260
3.86544589720640 6.32664176367285
3.61927294746144 6.27935930946325
3.95584762348697 3.89390320008437
4.53062506445827 2.28175233943933
4.19990452535833 2.15544301777705
4.35659188802714 0.08576502972247
5.87362160279331 -0.60637741205545
4.94488255274615 -2.25033187783735
4.02489210925304 -2.76893117120001
4.49939898211824 -3.92606280519643
3.54463330891794 -5.48048223753038
3.04299968801745 -7.60683470582086
3.26607455580647 -7.21014983536375

```
 3.51100552876526  -7.13440552148986
 2.48595502853656  -8.15201972726381
 2.23450327175463 -10.24600635846441
 0.54275781698829  -9.73302876290198
 0.35030306011783 -10.33014112914620
-0.15746397170300 -10.61473183445742
-1.35455740891787 -10.66175909993534
-2.03511163469000  -9.51254740004811
-1.28189225442613  -9.55534666161366
-2.40202143267810  -8.72958608633275
-3.32601375087707  -8.31089408379386
-2.93929825377602  -7.56867029451679
-3.71359060741068  -6.15166354717752
-4.81549543051928  -4.92129819085907
-3.72038384196387  -2.99049681378782
-4.58158136075272  -3.90454858960753
-4.02408601903468  -2.48737923245777
-4.30731683442634  -0.00520467606188
-5.40061679322316   0.39402546620703
-5.68659788062301   1.04505377144534
-4.79903834723396   3.27949552341289
-4.72249459026576   5.18973845675993
-4.72781621100295   5.78300428056945
-3.55390441536928   7.54867075057540
-2.52964124710135   7.59241168636941
-3.26681179832294   9.20888735604066
-2.91682326883842   8.55917973398965
-1.98842468027433   8.61389721749457
-0.84563494831073   9.55915752125815
-1.13918459250291  10.63529397489550
-0.72078888920343   9.33027474532863]
```

Singular Value Decomposition And Fitting

For a $m \times n$ matrix **A**, the SVD function of NewMat can decompose it into $A = UDV^T$, where

U and V are unitary matrices, and V=[v_1, v_2, …,v_n]; D is diagonal, and $D = \begin{bmatrix} \delta_1 & 0 & \cdots & 0 \\ 0 & \delta_2 & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_n \end{bmatrix}$,

Note: NewMat can't guarantee $a_1 \geq a_2 \geq \cdots \geq a_n$, while Matlab can.

For non-solution linear system $Ax = 0$, the solution by means of the least square is $x = v_i$, where $v_i = \min(\{\delta_m \mid m \in [1,n]\})$. Or, the column vector of V corresponding to the least singular value of D.

E.g. in the case of circle fitting, firstly, write a circle in the form of

$a_1(x^2 + y^2) + a_2 x + a_3 y + a_4 = 0$

Secondly, convert the data we have into matrix A, where

$$A=\begin{bmatrix} x^2{}_1+y^2{}_1 & x_1 & y_1 & 1 \\ x^2{}_2+y^2{}_2 & x_3 & y_2 & 1 \\ x^2{}_3+y^2{}_3 & x_4 & y_3 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x^2{}_{100}+y^2{}_{100} & x_{100} & y_{100} & 1 \end{bmatrix}$$

Thirdly, if δ_5 is minimum singular value of D, then $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = v_5$ is the solution.

Circle Fitting of Known Radius by Iteration

Question: Given a set of points: (X, Y), where X, Y are N by 1 vectors (elements of X are x_i, elements of Y are y_i. Actual radius is R, estimated origin is (x_c, y_c). The solution is a (x_c, y_c) to make the error then minimum.

Error= $\sum \left| \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - R \right|$

The gradient of Error is:

$$J=\begin{bmatrix} \frac{\partial Error}{\partial x_i} & \frac{\partial Error}{\partial y_i} \end{bmatrix} = \begin{bmatrix} \frac{(x_i - x_c)}{\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}} & \frac{(y_i - y_c)}{\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}} \end{bmatrix}$$

Then update *u* by:

$$Jh + Error = 0$$

$$\begin{bmatrix} x_c & y_c \end{bmatrix} = \begin{bmatrix} x_c & y_c \end{bmatrix} + h$$

Which would converge $x_c, y_c,$ to the best origin.

Index

A

algcircle, 4
AlgCircle, 5

C

circle, 5

F

fitellipa, 4
FitEllipse, 8

G

gradient, 13

K

krcircle, 4
KrCircle, 7

S

Singular Value Decomposition, 12
SolveEllipse, 10