计算机视觉

目　录

1 说明

这份文档主要关注对椭圆参数的基础矩阵估计过程。总结主要的实践与问题。按照原先课堂上的说明对基础矩阵估计的具体编程实现。主体过程如下：

1. 归一化，椭圆质心原点化以及放缩
2. SVD或者LS初始化，对椭圆的初始参数进行设定
3. Sampson error计算，以及使用Gauss-Newton迭代进行计算
4. 去归一化，坐标还原

2 椭圆的参数估计

通过给定的数据点，进行椭圆的参数估计并绘出椭圆。椭圆方程如下：

$$[x^2, y^2, xy, x, y, 1][a, b, c, d, e, f]^T = 0$$

需要根据给定的点坐标，对a, b, c, d, e, f进行估计。同时可以令常数项f为1。这样即需要估计5个参数的值。

通过构造数据点(x_i, y_i)的向量:

$$\boldsymbol{w}_i = [x_i^2, y_i^2, x_i y_i, x_i, y_i, 1]$$

所以所有点的生成的矩阵$\boldsymbol{W}$为：

$$\boldsymbol{W} = [\boldsymbol{w}_1^T; \boldsymbol{w}_2^T; \ldots; \boldsymbol{w}_n^T]^T$$

这样椭圆的参数估计问题变为求解方程组：

$$\boldsymbol{W}[a, b, c, d, e, f]^2 = 0$$

这个方程的解存在三种情况：

a 少于5个线性无关约束，意味着方程有无穷多解。

b 等于5个线性无关约束，存在唯一解。

c 大于5个线性无关约束，只存在近似解。

当大于5个线性无关约束情况下可以通过SVD或者LS进行一般估计，特别是在误差较小的情况下。

3 归一化

进行数据归一化，可以提高计算的精度。主要为两个过程：

- 中心平移：即估计当前椭圆的中心点(c_1, c_2)，对x和y 坐标进行移动。使得椭圆中心在原点上。一般(c_1, c_2)取椭圆的质心。
- 距离放缩：即对平移后的坐标进行放缩，具体如下，取$s = \sqrt{\left(\sum_{i=1}^{n} x_i^2 + y_i^2\right)/2n}$，$x = x/s$和$y = y/s$。

尽管，归一化一定程度上可以减小误差，但是在实际的计算中，影响更大的是点在椭圆的范围分布，如果点只在椭圆某一部分密集集中，一般估计效果都不好，而且通过质心估算的中心位置非常不准确，同样的放缩也会一样的不准确。

4 去归一化

因为归一化使得,x和y的坐标值发生了改变，但估计出最终的参数值时需要将各个参数还原到原有的坐标上。具体的还原转换式子如下：

$$
\begin{aligned}
&a = a_0/s^2 \\
&b = b_0/s^2 \\
&c = c_0/s^2 \\
&d = d_0/s - 2a_0 * c1/s^2 - c_0 * c2/s^2 \\
&e = e_0/s - 2 * b_0 * c2/s^2 - c_0 * c1/s^2 \\
&f = f_0 + a_0 * c1^2/s^2 + c_0 * c1 * c2/s^2 + b_0 * c2^2/s^2 - d_0 * c1/s - e_0 * c2/s
\end{aligned}
$$

这个式子在ppt上略有错误，参照上下文的话。因为一般的椭圆参数方程写写成：

$$[x^2, xy, y^2, x, y, 1][a, b, c, d, e, f]^T = 0$$

而ppt上的参数格式是基于：

$$[x^2, y^2, xy, x, y, 1][a, b, c, d, e, f]^T = 0$$

这两者还有有些区别的，特别是在去归一化上，参数的b和c的位置不同。这样因为位置不同而表达方式不同之处，有不少地方都出现了，这篇文章全是基于第二个表达方程的。

5 SVD初始化

首先需要说明为什么SVD可以进行近似估计和如何选取最终估计值。由第2节的参数说明可以知道：

$$\boldsymbol{W}[a,b,c,d,e,f]^2=0$$

需要找到一组近似解来对a,b,c,d,e,f进行初始化赋值。SVD可以很好的完成这个任务。目标函数为：

$$\min \quad \left(\sum_{i=1}^{n}\boldsymbol{W}_i\boldsymbol{x}\right)^2$$

展开形式为：

$$\min \quad \boldsymbol{x^T}\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{x}$$

令$\boldsymbol{M}=\boldsymbol{W}^T\boldsymbol{W}$，且规范化令

$$\|\boldsymbol{x}\|=1$$

。原始目标函数变为：

$$\min \quad \boldsymbol{x^T}\boldsymbol{M}\boldsymbol{x}$$
$$s.t. \quad \|\boldsymbol{x}\|=1$$

使用拉格朗日乘数法，令

$$f(x)=\boldsymbol{x^T}\boldsymbol{M}\boldsymbol{x}-\lambda(\boldsymbol{x^T}\boldsymbol{x}-1)=0$$

求偏导：

$$\frac{\partial f}{\partial \boldsymbol{x}}=\boldsymbol{M}\boldsymbol{x}+\lambda\boldsymbol{x}=0$$

即得：

$$\boldsymbol{M}\boldsymbol{x}=\lambda\boldsymbol{x}$$

又由SVD分解可得：

$$\boldsymbol{M}=\boldsymbol{V}\boldsymbol{\Lambda^2}\boldsymbol{V^T}$$

取奇异值最小的对于的V的向量得到：

$$\boldsymbol{M}\boldsymbol{v_k}={\sigma_k}^2\boldsymbol{u_k}$$

使得目标函数得到近似的最小值，σ_k为最小奇异值。这样可以使得乘积逼近零。

或者直接用$\boldsymbol{W}\boldsymbol{v_k}=\sigma_k\boldsymbol{u_k}$也是成立的。但是一般$x$的估计值与这里的不同，当然如果仅作为迭代的初始值，也足够了。

6 Sampson error

问题即最小化真实值与实际值之间的差别。即：

$$\min_{f(\overset{\wedge}{\boldsymbol{x}}_{\boldsymbol{i}})=0} \sum \left\| \boldsymbol{x_i} - \overset{\wedge}{\boldsymbol{x}}_{\boldsymbol{i}} \right\|^2$$

要最小化目标函数，得到最优的系数解。

一阶泰勒展开：

$$f(\boldsymbol{x}+\delta\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{\partial f}{\partial \boldsymbol{x}}\delta\boldsymbol{x}$$

又$\overset{\wedge}{\boldsymbol{x}} = \boldsymbol{x} + \delta\boldsymbol{x}$ 且$f(\overset{\wedge}{\boldsymbol{x}}) = 0$ 由一阶展开式可得：

$$\boldsymbol{J^T}\delta\boldsymbol{x} = -\varepsilon$$

其中$\boldsymbol{J} = \frac{\partial f}{\partial \boldsymbol{x}}$，$\varepsilon = f(\boldsymbol{x})$ 原问题可以转化为：

$$\min_{\boldsymbol{J}^T\delta\boldsymbol{x}=-\varepsilon} \|\delta\boldsymbol{x}\|^2$$

使用拉格朗日乘数法带入：

$$\|\delta\boldsymbol{x}\|^2 + 2\lambda(\boldsymbol{J}^T\delta\boldsymbol{x} + \varepsilon)$$

对$\boldsymbol{x}$求偏导，取偏导为零：

$$\delta\boldsymbol{x} + \lambda\boldsymbol{J} = \boldsymbol{0}$$

可得：

$$\delta\boldsymbol{x} = -\lambda\boldsymbol{J}$$

带回$\boldsymbol{J}^T\delta\boldsymbol{x} = -\varepsilon$可得：

$$\lambda = (\boldsymbol{J}^T\boldsymbol{J})^{-1}\varepsilon$$

带回$\delta\boldsymbol{x} + \lambda\boldsymbol{J} = \boldsymbol{0}$,解之得：

$$\delta\boldsymbol{x} = -\varepsilon(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}$$

所以，

$$\|\delta\boldsymbol{x}\|^2 = (\delta\boldsymbol{x})^T\delta\boldsymbol{x} = (\varepsilon^2(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J})^T(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}$$

化简：

$$\|\delta\boldsymbol{x}\|^2 = \frac{\|\varepsilon\|^2}{\boldsymbol{J}^T\boldsymbol{J}}$$

最终Sampson error为：

$$\sum_i \frac{\|\varepsilon_i\|^2}{\boldsymbol{J}_i{}^T\boldsymbol{J}_i}$$

7 Gauss-Newton迭代

计算Sampson error的具体表达形式：

$$\sum_i \frac{\|\varepsilon_i\|^2}{\boldsymbol{J}_i{}^T\boldsymbol{J}_i}$$

由定义，$f(x)=\varepsilon$且$f(x)=\boldsymbol{W}\boldsymbol{p}$所以可得：

$$\|\varepsilon_i\|^2=\boldsymbol{p}^T\boldsymbol{w}_i^T\boldsymbol{w}_i\boldsymbol{p}$$

且令$\boldsymbol{A}_i=\boldsymbol{w}_i^T\boldsymbol{w}_i$，这里的$\boldsymbol{p}=[a,b,c,d,e,f]^T$。

继续求解J_i表达式：

$$f(x,y)=ax^2+by^2+cxy+dx+ey+f$$

分别对x和y求偏导得到：

$$\boldsymbol{J}_i=\begin{bmatrix}\frac{\partial f(x,y)}{\partial x}\\ \frac{\partial f(x,y)}{\partial y}\end{bmatrix}$$

所以可得：

$$\boldsymbol{J}_i=\begin{bmatrix}2ax+cy+d\\ 2by+cx+e\end{bmatrix}=\begin{bmatrix}2x&0&y&1&0&0\\ 0&2y&x&0&1&0\end{bmatrix}[a,b,c,d,e,f]^T$$

于是令：

$$\boldsymbol{B}_i=\begin{bmatrix}2x&0&y&1&0&0\\ 0&2y&x&0&1&0\end{bmatrix}^T\begin{bmatrix}2x&0&y&1&0&0\\ 0&2y&x&0&1&0\end{bmatrix}$$

可将原Sampson error化为：

$$obj=\sum_i\frac{\boldsymbol{p}^T\boldsymbol{A}_i\boldsymbol{p}}{\boldsymbol{p}^T\boldsymbol{B}_i\boldsymbol{p}}$$

进而表示为2范式形式为：

$$obj=\sum_i\left(\frac{\boldsymbol{w}_i\boldsymbol{p}}{\sqrt{\boldsymbol{p}^T\boldsymbol{B}_i\boldsymbol{p}}}\right)^2$$

对目标函数进行一阶偏导展开:

$$\nabla_{\boldsymbol{p}}obj=2\sum_i\left(\frac{\boldsymbol{w_i}\boldsymbol{p}}{\sqrt{\boldsymbol{p}^TB_i\boldsymbol{p}}}\right)\left(\frac{(\boldsymbol{w_i}\boldsymbol{p})'}{\sqrt{\boldsymbol{p}^T\boldsymbol{B_i}\boldsymbol{p}}}-\frac{1}{2}\frac{\boldsymbol{w_i}\boldsymbol{p}}{\left(\sqrt{\boldsymbol{p}^T\boldsymbol{B_i}\boldsymbol{p}}\right)^3}\left(\boldsymbol{p}^T\boldsymbol{B_i}\boldsymbol{p}\right)'\right)$$

由前文的定义可以知道：$\boldsymbol{w_i}\boldsymbol{p}$和$\boldsymbol{p}^T\boldsymbol{B_i}\boldsymbol{p}$均为1*1矩阵，需要对向量$\boldsymbol{p}$进行求导。可以使用如下几个公式：

$$tr\ a=a$$

$$tr\ \boldsymbol{AB}=tr\ \boldsymbol{BA}$$

$$\nabla_{\boldsymbol{A}^T}f(\boldsymbol{A})=(\nabla_{\boldsymbol{A}}f(\boldsymbol{A}))^T$$

$$\nabla_{\boldsymbol{A}}tr\ \boldsymbol{AB}=\boldsymbol{B}^T$$

$$\nabla_{\boldsymbol{A}}tr\ \boldsymbol{ABA}^T\boldsymbol{C}=\boldsymbol{CAB}+\boldsymbol{C}^T\boldsymbol{AB}^T$$

所以对于$\boldsymbol{w_i p}'$得：

$$(\boldsymbol{w_i p})' = \nabla_{\boldsymbol{p}}\, tr\, \boldsymbol{w_i p} = \boldsymbol{w_i}^T$$

同样对于$\left(\boldsymbol{p}^T \boldsymbol{B_i p}\right)'$

$$\left(\boldsymbol{p}^T \boldsymbol{B_i p}\right)' = (\nabla_{\boldsymbol{p}^T} tr\, \boldsymbol{p}^T \boldsymbol{B_i p} I)^T = (\boldsymbol{I p}^T \boldsymbol{B_i} + \boldsymbol{I p}^T \boldsymbol{B_i}^T)^T = \boldsymbol{B_i}^T \boldsymbol{p} + \boldsymbol{B_i p}$$

又$\boldsymbol{B_i}$为对称矩阵，所以化简为：

$$\left(\boldsymbol{p}^T \boldsymbol{B_i p}\right)' = 2\boldsymbol{B_i p}$$

带回原始目标函数一阶偏导可得：

$$\nabla_{\boldsymbol{p}} obj = 2\sum_i \left(\frac{\boldsymbol{w_i p}}{\sqrt{\boldsymbol{p}^T \boldsymbol{B_i p}}}\right)\left(\frac{\boldsymbol{w_i}^T}{\sqrt{\boldsymbol{p}^T \boldsymbol{B_i p}}} - \frac{\boldsymbol{w_i p}}{\left(\sqrt{\boldsymbol{p}^T \boldsymbol{B_i p}}\right)^3}\boldsymbol{B_i p}\right)$$

有文献[1] 可得到Gauss-Newton Hessian矩阵的表示形式：

$$\begin{aligned} & f(\theta) = \textstyle\sum_i f_i(\boldsymbol{x}_i;\theta) = r^2(\boldsymbol{x}_i;\theta) \\ & H_{GN} = 2\textstyle\sum \frac{\partial \boldsymbol{r}_i}{\partial\theta}\left(\frac{\partial \boldsymbol{r}_i}{\partial\theta}\right)^T \end{aligned}$$

在本文中，令：

$$\nabla_{\boldsymbol{p}} obj = 2\sum_i \left(\frac{\boldsymbol{w_i p}}{\sqrt{\boldsymbol{p}^T \boldsymbol{B_i p}}}\right)\left(\frac{\boldsymbol{w_i}^T}{\sqrt{\boldsymbol{p}^T \boldsymbol{B_i p}}} - \frac{\boldsymbol{w_i p}}{\left(\sqrt{\boldsymbol{p}^T \boldsymbol{B_i p}}\right)^3}\boldsymbol{B_i p}\right) = 2\sum_i \left(\frac{\boldsymbol{w_i p}}{\sqrt{\boldsymbol{p}^T \boldsymbol{B_i p}}}\right)\boldsymbol{b_i}$$

其中$\boldsymbol{b_i} = \frac{\boldsymbol{w_i}^T}{\sqrt{\boldsymbol{p}^T \boldsymbol{B_i p}}} - \frac{\boldsymbol{w_i p}}{\left(\sqrt{\boldsymbol{p}^T \boldsymbol{B_i p}}\right)^3}\boldsymbol{B_i p}$ 所以Gauss-Newton Hessian矩阵形式为：

$$H_{GN} = 2\sum_i \boldsymbol{b_i b_i}^T$$

梯度向量

$$\boldsymbol{b} = \nabla_{\boldsymbol{p}} Obj = 2\sum_i \left(\frac{\boldsymbol{w_i p}}{\sqrt{\boldsymbol{p}^T \boldsymbol{B_i p}}}\right)\boldsymbol{b_i}$$

所以参数$\boldsymbol{p}$的更新方程为：

$$\boldsymbol{p} = \boldsymbol{p} - H^{-1}\boldsymbol{b}$$

8 编码与实现

编码主要通过人为设定长轴以，短轴，椭圆中心以及旋转角度等生成椭圆数据，同时加入高斯噪声。之后再用之前提到的方法使用SVD初始化进行牛顿迭代计算出最终的椭圆拟合方程。

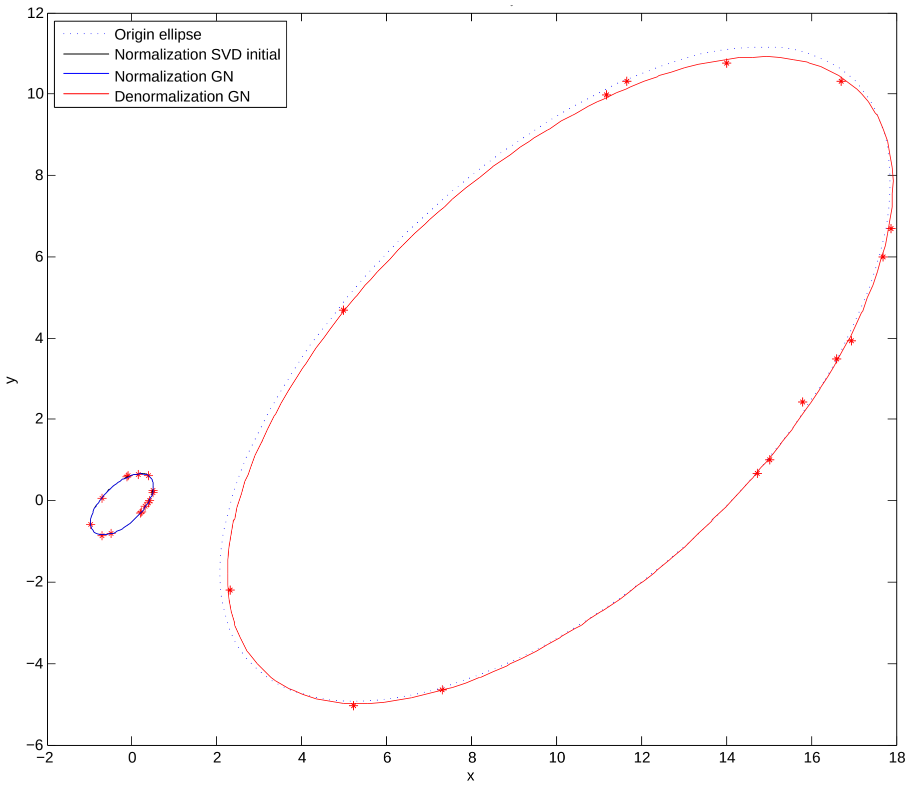

图 1 ABC轨迹和碰面示例

如图1所示，使用15个点作出的椭圆。使用文件graph.m即可生成这幅图像。

参考文献

[1] P. Chen, Hessian matrix vs. Gauss-Newton Hessian matrix, SIAM J. Numerical Analysis, vol. 49, no. 4, pp. 1417-1435, 2011.

[2] 李航.统计学习方法.北京:清华大学出版社.2012.p218-219

1 实现代码

生成x和y轴的高斯噪声。二者是独立的。

```
n = 40;
 X = normrnd(0,0.1,[n,1]);
fid = fopen('normrndxdata','w');
for i = 1:1:n
    fprintf(fid, '%f ',X(i));
end
fclose(fid);
Y = normrnd(0,0.1,[n,1]);
fid = fopen('normrndydata','w');
for i = 1:1:n
    fprintf(fid, '%f ',Y(i));
end
fclose(fid);
```

生成椭圆点的随机数据。

```
n = 40;
% 设置随机数种子
ctime = datestr(now, 30);
tseed = str2double(ctime((length(ctime) - 5) : end)) ;
rng(tseed);
T = rand(n,1);
fid = fopen('randdata','w');
for i = 1:1:n
    fprintf(fid, '%f ',T(i));
end
fclose(fid);
```

生成椭圆数据，加入噪声。

```
n = 40; % 取样点个数
A = zeros(n,2);
angle =2 *pi;
theta = 45;
R = [cosd(theta) -sind(theta);
```

```
            sind(theta)  cosd(theta)];  % 旋转矩阵
a = 10; % 长轴长
b = 5; % 短轴长

Ox =2;
Oy = 3;
% 读入随机数，省的每次生成的不同。固定随机数
 fid  = fopen('randdata','rt');
T= fscanf(fid, '%f',[n,1]);
fclose(fid);
% 读入随机数，省的每次生成的不同。固定随机数
 fid  = fopen('normrndxdata','rt');
X= fscanf(fid, '%f',[n,1]);
fclose(fid);
% 读入随机数，省的每次生成的不同。固定随机数
 fid  = fopen('normrndydata','rt');
Y= fscanf(fid, '%f',[n,1]);
fclose(fid);
%要先旋转后平移，才能保证和的数据一致。可能跟函数的具体实现过程相关ellipse1
for i = 1:1:n
    t = T(i) *angle;
    x = a*sin(t) + X(i);
    y = b*cos(t) + Y(i) ;
    A(i,:) =(R*[x;y])';
    A(i,1)=A(i,1)+Ox;
    A(i,2)=A(i,2)+Oy;
end
 plot(A(:,1),A(:,2),'r*');
 fid  = fopen('eclipse_data','w');
fprintf(fid, '%f %f ',A(:,1),A(:,2));
fclose(fid);
hold on
 ecc = axes2ecc(a,b); % 根据长半轴和短半轴计算椭圆偏心率
 [elat,elon]  = ellipse1(Ox,Oy,[a ecc],theta);
```

```
  col=plot(elat,elon,'c-')
 hleg=legend(col,'Origin ellipse')
 set(hleg,'Location','NorthWest')
```

使用SVD初始化，并用Gaussian-Newton迭代，绘制图片。

```
n = 15; % 取样点个数
A = zeros(n,2);
angle =2 *pi;
theta = 45;
R = [cosd(theta) -sind(theta);
          sind(theta)  cosd(theta)];  % 旋转矩阵
a = 10; % 长轴长
b = 5; % 短轴长

Ox =10;
Oy = 3;
% 读入随机数，省的每次生成的不同。固定随机数
 fid  = fopen('randdata','rt');
T= fscanf(fid,  '%f',[n,1]);
fclose(fid);
% 读入随机数，省的每次生成的不同。固定随机数
 fid  = fopen('normrndxdata','rt');
X= fscanf(fid,  '%f',[n,1]);
fclose(fid);
% 读入随机数，省的每次生成的不同。固定随机数
 fid  = fopen('normrndydata','rt');
Y= fscanf(fid,  '%f',[n,1]);
fclose(fid);
%要先旋转后平移，才能保证和的数据一致。可能跟函数的具体实现过程相关ellipse1
for  i  = 1:1:n
    t  = T(i) *angle;
    x = a*sin(t) + X(i);
    y = b*cos(t) + Y(i) ;
    A(i,:) =(R*[x;y])';
    A(i,1)=A(i,1)+Ox;
```

```
    A(i,2)=A(i,2)+Oy;
end
 plot(A(:,1),A(:,2),'r*');
 fid  = fopen('eclipse_data','w');
fprintf(fid, '%f %f ',A(:,1),A(:,2));
fclose(fid);
hold on
 ecc = axes2ecc(a,b); % 根据长半轴和短半轴计算椭圆偏心率
 [elat,elon] = ellipse1(Ox,Oy,[a ecc],theta);
  col=plot(elat,elon,'b:')
 hleg=legend(col,'Origin ellipse')
 set(hleg,'Location','NorthWest')
% 读入随机数，省的每次生成的不同。固定随机数
 fid  = fopen('eclipse_data','rt');
D = fscanf(fid, '%f %f',[n,2]);
fclose(fid);
hold on
 plot(D(:,1),D(:,2),'r*');
DN = zeros(n,2);

% 计算平均距离和，进行过程 SNormalization
x_mean = 0;
y_mean = 0;
dist_mean = 0;
for i = 1:1:n
    x_mean = x_mean + D(i,1);
    y_mean = y_mean + D(i,2);
    dist_mean = dist_mean + D(i,1)^2+D(i,2)^2;
end
x_mean = x_mean /n;
y_mean = y_mean/n;
dist_mean =sqrt( dist_mean/(2*n));
for i = 1:1:n
    D(i,1)=(D(i,1) - x_mean)/dist_mean;
```

```
    D(i,2) =(D(i,2) - y_mean)/dist_mean;
end
hold on
plot(D(:,1),D(:,2),'r*');

 W = zeros(n,6);
for i = 1:1:n
    xi = D(i,1);
    yi = D(i,2);
    w= [xi^2,yi^2,xi*yi,xi,yi,1];
   W(i,:) =w;
end
[u,s ,v] = svd(W);
v6 = v(:,6);
a = v6(1);b=v6(2);c=v6(3);d=v6(4);e=v6(5);f=v6(6);
hold on
 str = sprintf('%.8f*x^2 %.8f*y^2+ %.8f*x*y +%.8f*x+ %.8f*y+ %.8f = 0',a,b,c,d,e,f);
 col1 = ezplot(str,[-10,10]);
 set(col1 ,'Color','k')
p = v6;
H = zeros(6,6);
iterate = 1000;
Obj = 0;
for it = 1:1:iterate
    Obj1 = Obj;
    for i = 1:1:n
        wi = W(i,:);
        Ai = p'*wi'*wi*p;
    j = [D(i,1)*2, 0, D(i,2),1, 0, 0;
           0, D(i,2)*2, D(i,1), 0, 1, 0];
     Ji = j*p;
     Bi = j'*j;
     De = sqrt(p'*Bi*p);
     bi = wi'/De - wi*p*Bi*p/(De^3);
```

```
        H = H + 2*bi*bi';
        b = b + wi*p*bi/De;
        Obj = Obj + (wi*p)^2/De;
    end

    for i = 1:1:6
        for k = 1:1:6
            if(H(i,k) < 0)
                H(i,k) = 0;
            end
        end
    end
    HI = inv(H);
    p = p - HI*b
    if (abs(Obj-Obj1) < 0.0001) % 两次迭代目标值改变量
        break;
    end

end

a = p(1);b=p(2);c=p(3);d=p(4);e=p(5);f=p(6);

hold on
str = sprintf('%.8f*x^2+%.8f*y^2+ %.8f*x*y +%.8f*x+ %.8f*y+ %.8f = 0',a,b,c,d,e,f);
col2 = ezplot(str,[-10,10]);
 set(col2 ,'Color','b')

s = dist_mean;
c1=x_mean;
c2=y_mean;
a = p(1);b=p(2);c=p(3);d=p(4);e=p(5);f=p(6);
% 上系数次序与之前的略有不同就是，和有pptxyy谁排第
    二。^2上ppt DE-normalization 是x^2+xy+y^2+x+y+f
% 这次编程使用的顺序是 xi^2,yi^2,xi*yi,xi,yi,1
```

```
a0 = a/s^2;
b0 = b/s^2;
c0 =c/s^2;
d0 = d/s-(2*a*c1)/s^2 - (c*c2)/s^2;
e0 = e/s -(2*b*c2)/s^2 - (c*c1)/s^2;
f0 = f + (a*c1^2)/s^2 + (c*c1*c2)/s^2+(b*c2^2)/s^2-(d*c1)/s-(e*c2)/s;
str0 = sprintf('%.8f*x^2+%.8f*y^2+ %.8f*x*y +%.8f*x+ %.8f*y+ %.8f = 0',a0,b0,c0,d0,e0,f0);
 col3 = ezplot(str0,[-40,40]);
 set(col3 ,'Color','r')
hleg=legend([col,col1,col2,col3], 'Origin ellipse', 'Normalization SVD initial','Normalization GN','Denormalization GN')
set(hleg,'Location','NorthWest')
```