基于张正友标定法的手机相机标定

赵思源
南京理工大学 机械工程学院 机器人工程 919101960141

摘要：在手机相机的成像系统中，为确定空间物体表面某点的三维几何位置与其在图像中对应点之间的相互关系，必须建立相机成像的几何模型，并求出相机的内外参数以及畸变系数。本文参照张正友标定法，首先对标定板上棋盘格角点进行检测与提取，再对世界坐标系与相机坐标系进行齐次转化，通过 Levenberg-Marquardt 优化算法求得内外参矩阵之积，再依据每张图片提供的两个约束关系，使用 SVD 分解求得相机内参矩阵，并综合考虑成像时的径向和切向畸变，获取图片的畸变系数。最终获得手机相机内参矩阵为 A （详见附录）,径向畸变系数为 $(-0.9823, 0.5366)$，切向畸变系数为 $\left(1.5740\times10^{-10}, 1.9018\times10^{-10}\right)$，拟合残差为 4.2815 （型号为 iPhone12 mini）。与 MATLAB 工具箱标定结果对比发现误差较小，满足了标定的精度需求，实现了手机相机的标定功能。
关键词：相机标定、齐次变换、最大似然估计、畸变

Mobile phone camera calibration based on Zhang Zhengyou's calibration method

Abstract: In the imaging system of the mobile phone camera, in order to determine the relationship between the three-dimensional geometric position of a point on the surface of the space object and its corresponding point in the image, the geometric model of the camera imaging must be established, and the internal and external parameters and distortion of the camera must be obtained. coefficient. Referring to Zhang Zhengyou's calibration method, this paper first detects and extracts the checkerboard corner points on the calibration board, then transforms the world coordinate system and the camera coordinate system homogeneously, and obtains the product of the internal and external parameter matrices through the Levenberg-Marquardt optimization algorithm. For the two constraints provided by each image, use SVD decomposition to obtain the camera's internal parameter matrix, and comprehensively consider the radial and tangential distortion during imaging to obtain the distortion coefficient of the image. Finally, the internal parameter matrix of my mobile phone camera is obtained as A,Radial distortion factor is $(-0.9823, 0.5366)$,Tangential distortion factor is $\left(1.5740\times10^{-10}, 1.9018\times10^{-10}\right)$ The fitting error is 4.2815 (model is iPhone12 mini). Compared with the calibration results of the MATLAB toolbox, it is found that the error is small, which meets the accuracy requirements of the calibration and realizes the calibration function of the mobile phone camera.
Keywords: camera calibration, homogeneous transformation, maximum likelihood estimation, distortion.

1.引言

机器人视觉的基本任务之一是从摄像机获取的图像信息出发计算三维空间中物体的几何信息，并由此重建和识别物体，而空间物体表面某点的三维几何位置与其

在图像中对应点之间的相互关系是由摄像机成像的几何模型决定的，这些几何模型参数就是摄像机参数。在大多数条件下，这些参数必须通过实验与计算才能得到。无论是在图像测量或者机器视觉应用中，相机参数的标定都是非常关键的环节，其标定结果的精度及算法的稳定性直接影响相机工作产生结果的准确性。因此，做好相机标定是做好后续工作的前提，提高标定精度是科研工作的重点所在。

张正友标定法是相机标定的一大经典算法，该标定法使用二维方格组成的标定板进行标定，采集标定板不同位姿图片，提取图片中角点像素坐标，通过单应矩阵计算出相机的内外参数初始值，利用非线性最小二乘法估计畸变系数，最后使用极大似然估计法优化参数。该方法操作简单，而且精度较高，可以满足大部分场合。本文对张正友标定法进行一定的改进，引入切向畸变，建立了更为完善的相机成像模型并矫正了透镜的畸变。

2.流程

2.1 符号说明

将世界坐标系下，三维点的坐标记为：

$$M=\left[X_W,Y_W,Z_W\right]^T$$

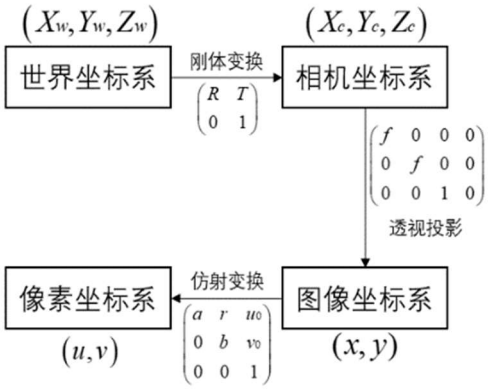
像素坐标系下,对应二维点的坐标记为：

$$m=\left[u,v\right]^T$$

并将两者扩大维数后的形式记为：

$$\widetilde{M}=\left[X_w,Y_w,Z_w,1\right]^T,\quad \widetilde{m}=\left[u,v\right]^T$$

因此，相机成像的基本流程如下图所示：



将表示刚体变换的矩阵记为外参矩阵 A,表示透视投影的矩阵记为内参矩阵，内外参矩阵的乘积记为 $H=\left[H_1,H_2,H_3\right]$。具体形式为

$$H=\begin{pmatrix}H_{11} & H_{12} & H_{13}\\ H_{21} & H_{22} & H_{23}\\ H_{31} & H_{32} & H_{33}\end{pmatrix}$$

2.2 求解 H

由于标定时，世界坐标系被看作固定在棋盘格上，因此棋盘格上任意一点满足 $Z_W=0$，若使用 R_1,R_2 表示内参矩阵中旋转矩阵 R 的前两列，则世界坐标系下点可以表示为 $\widetilde{M}=\left[X_C,Y_C,1\right]^T$,完成对世界坐标系和像素坐标系的齐次变换，若将尺度因子记为 Z，则成像原理可被写为如下形式：

$$Z\begin{pmatrix}u\\ v\\ 1\end{pmatrix}=\begin{pmatrix}a & r & u_0\\ 0 & b & v_0\\ 0 & 0 & 1\end{pmatrix}\left(R_1,R_2,T\right)\begin{pmatrix}X_W\\ Y_W\\ 1\end{pmatrix}$$

因此，可以得出像素坐标系与世界坐标系之间的关系：

$$u=\frac{H_{11}X+H_{12}Y+H_{13}}{H_{31}X+H_{32}Y+H_{33}}$$

$$v=\frac{H_{21}X+H_{22}Y+H_{33}}{H_{31}X+H_{32}Y+H_{33}}$$

H 为齐次矩阵，具有 8 个独立元素，因此只需要一张图片上的角点个数等于 4 时，即可完成 H 矩阵的求解，但此仅使用 4 个角点求解会导致误差较大，虽可以使用最小二乘法估计得到的解，但结果并无实际意义，为进一步增强标定结果的可靠性，参照张正友标定法附录 A，使用最大似然估计的方法求解

设

$$x=\left[H_1,H_2,H_3\right]$$

$$L=\begin{pmatrix}\widetilde{M}^T & 0^T & -u\widetilde{M}^T\\ 0^T & \widetilde{M}^T & -v\widetilde{M}^T\end{pmatrix}$$

则前文所提及的成像原理关系式可写为

$$Lx = 0$$

若使用 $\bar{h}_i$ 表示 H 矩阵的第 i 行，并将 $\hat{m}_i$ 记为:

$$\hat{m}_i = \frac{1}{\bar{h}_3^T M_i}\begin{bmatrix}\bar{h}_1^T M_i \\ \bar{h}_2^T M_i\end{bmatrix}$$

此条件下，使用 Levenberg-Marquardt 算法最小化函数

$$f(x) = \sum_i (m_i - \hat{m}_i)^T \sigma^2 I (m_i - \hat{m}_i)$$

即可得到 H 的最大似然估计值

2.3 求解 A

2.3.1 内参的约束关系

根据前文假设 R_1, R_2 为旋转矩阵的前两列，根据 刚性变换中旋转矩阵的约束关系，两列均为单位向量且相互垂直，可表示为如下形式:

$$R_1^T R_2 = 0$$
$$R_1^T R_1 = R_2^T R_2 = 1$$

将 $R_1 = A^{-1}H_1, R_2 = A^{-1}H_2$ 代入可得:

$$H_1^T A^{-T} A^{-1} H_2 = 0$$
$$H_1^T A^{-T} A^{-1} H_1 = H_2^T A^{-T} A^{-1} H_2 = 1$$

2.3.2 求解 B

观察可知，若想要求解 A ,可以选择添加中间变量 $B = A^{-T}A^{-1}$, B 为对称矩阵，此时

$$HBH = \begin{pmatrix}H_{1i}\\H_{2i}\\H_{3i}\end{pmatrix}^T \begin{pmatrix}B_{11} & B_{12} & B_{13}\\B_{21} & B_{22} & B_{23}\\B_{31} & B_{32} & B_{33}\end{pmatrix}\begin{pmatrix}H_{1j}\\H_{2j}\\H_{3j}\end{pmatrix}$$

$$= \begin{pmatrix}H_{1i}H_{1j}\\H_{1i}H_{2j} + H_{2i}H_{1j}\\H_{2i}H_{2j}\\H_{1i}H_{3j} + H_{3i}H_{1j}\\H_{2i}H_{3j} + H_{3i}H_{2j}\\H_{3i}H_{3j}\end{pmatrix}^T \begin{pmatrix}B_{11}\\B_{12}\\B_{22}\\B_{13}\\B_{23}\\B_{33}\end{pmatrix}$$

将上式左边部分记为 v_{ij} ，右边部分记为 b ，则 R_1, R_2 均为单位向量且相互正交的条件可写为如下形式:

$$\begin{pmatrix}v_{12}\\v_{11} - v_{22}\end{pmatrix} b = vb = 0$$

由此，可以通过构建 v 矩阵的方式，利用 $vb = 0$ 的约束条件求得 B 矩阵。

2.3.3 利用 B 求解 A

由于 $B = A^{-T}A^{-1}$ ，且 A 由 a, r, b, u_0, v_0 五个参数构成，故 B 可写为如下形式:

$$B = \begin{pmatrix}\frac{1}{a^2} & -\frac{r}{a^2 b} & \frac{rv_0 - bu_0}{a^2 b}\\ -\frac{r}{a^2 b} & \frac{1}{b^2} + \frac{r^2}{a^2 b^2} & \frac{r(bu_0 - rv_0)}{a^2 b^2} - \frac{v_0}{b^2}\\ \frac{rv_0 - bu_0}{a^2 b} & \frac{r(bu_0 - rv_0)}{a^2 b^2} - \frac{v_0}{b^2} & \frac{(bu_0 - rv_0)^2}{a^2 b^2} + \frac{v_0^2}{b^2} + 1\end{pmatrix}$$

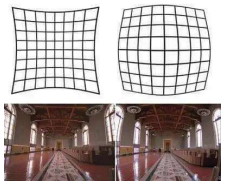
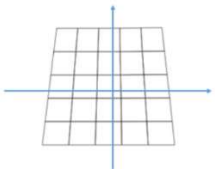
进而可求得相机内参:

$$V_0 = \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2}$$
$$a = \sqrt{\frac{1}{B_{11}}}$$
$$b = \sqrt{\frac{B_{11}}{B_{11}B_{22} - B_{12}^2}}$$
$$\mathrm{r} = -B_{12}a^2 b$$
$$u_0 = \frac{rv_0}{b} - B_{13}a^2$$

2.4 求解畸变参数

由透镜形状引起的畸变称为径向畸变，径向畸变主要分为桶形畸变和枕型畸变。在针孔模型中，一条直线投影到像素平面上还是一条直线。 但在实际中，相机的透镜往往使得真实环境中的一条直线在图片中变成了曲线。越靠近图像的边缘现象越明显。 由于透镜往往是中心对称的，这使得不规则畸变通常径向对称。

除了透镜的形状会引入径向畸变，在相机的组装过程中由于不能使透镜严格和成像平面平行，会引入切向畸变。

将像平面上真实观测的坐标记为$(\hat{x},\hat{y})$（含有畸变），校正后点的坐标记为(x,y)（无畸变），$r=\sqrt{x^2+y^2}$。

由于径向畸变随着与中心距离的增大而增大，可以使用与距离有关的多项式进行修正：

$$\hat{x}=x(1+k_1r^2+k_2r^4)$$
$$\hat{y}=y(1+k_1r^2+k_2r^4)$$

其中，k_3主要对畸变很大的相机起作用，对于普通相机可忽略其影响。

对于切向畸变，可以使用另外两个参数进行修正：

$$\hat{x}=x+k_3(r^2+2x^2)+2k_4xy$$
$$\hat{y}=y+k_3(r^2+2y^2)+2k_4xy$$

将上述两式子合并，即可得到：

$$\hat{x}=x(1+k_1r^2+k_2r^4)+k_3(r^2+2x^2)+2k_4xy$$
$$\hat{y}=y(1+k_1r^2+k_2r^4)+k_3(r^2+2y^2)+2k_4xy$$

最终得到 4 个畸变参数(k_1,k_2,k_3,k_4)。在求得畸变参数后，即可对图像进行畸变校正。与推断H矩阵的方法类似，通过最大似然估计的方法以下式子的最小值：

$$f(x)=\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\hat{m}_{i,j}-m(k_1,k_2,k_3,k_4,F,R_i,T_i,M_{i,j})\right\|$$

通过 Levenberg-Marquardt 算法即可求得畸变参数的非线性解，进而完成图片的畸变校正工作。

3.实验结果


以两张图片为例，首先求得其内外参矩阵之积：

$$H_1=\begin{pmatrix}11.1 & 0.2 & 901.8\\ 0 & 9.2 & 1222.7\\ 0 & 0 & 1\end{pmatrix}$$

$$H_2=\begin{pmatrix}0.2 & 10.2 & 940.2\\ -9 & -0.7 & 2567.5\\ 0 & 0 & 1\end{pmatrix}$$

结合数据集中其余图片的H矩阵，即可求得$B=A^{-T}A^{-1}$：

$$B=\begin{pmatrix}5.534\times10^{-8} & -3.3761\times10^{-11} & 5.5361\times10^{-8}\\ -3.3761\times10^{-11} & -8.6135\times10^{-5} & -1.1178\times10^{-4}\\ 5.5361\times10^{-8} & -1.1178\times10^{-4} & 1\end{pmatrix}$$

进而求得内参矩阵A：

$$A=\begin{pmatrix}3400.7 & 2.0743 & 1556.5\\ 0 & 3400.1 & 2020.2\\ 0 & 0 & 1\end{pmatrix}$$

将此矩阵视作无误差的理想情况，即可求得图片对应的简化外参矩阵：

$$RT_1=\begin{pmatrix}0.9890 & 0.1016 & -61.4779\\ -0.0784 & 0.9764 & -74.9485\\ 0.1254 & -0.1904 & 319.5618\end{pmatrix}$$

$$RT_2=\begin{pmatrix}-0.0623 & 0.9989 & -59.8562\\ -0.09825 & -0.0653 & 53.1375\\ 0.1754 & -0.0115 & 330.1283\end{pmatrix}$$

再次使用 L-M 算法，求解优化后的内参矩阵A和畸变参数：

$$A=\begin{pmatrix}3329.2 & 2.7 & 1524.4\\ 0 & 3323.0 & 1991.8\\ 0 & 0 & 1\end{pmatrix}$$

$$\begin{pmatrix}k_1\\ k_2\\ k_3\\ k_4\end{pmatrix}=\begin{pmatrix}-0.9823\\ 0.5366\\ 1.5740\times10^{-10}\\ 1.1908\times10^{-10}\end{pmatrix}$$

同时求得拟合残差为4.2815。

参考文献

[1]冯亮,谢劲松,李根,霍庆立.摄像机标定的原理与方法综述[J].机械工程师,2016(01):18-20.

[2]刘艳,李腾飞.对张正友相机标定法的改进研究[J].光学技术,2014,40(06):565-570.DOI:10.13741/j.cnki.11-1879/o4.2014.06.017.

[3]舒娜. 摄像机标定方法的研究[D].南京理工大学,2014.

[4]杨雪荣,张湘伟,成思源,黄曼慧.视觉测量中的相机标定方法进展研究[J].机械设计与制造,2009(03):259-261.

[5]毛剑飞,邹细勇,诸静.改进的平面模板两步法标定摄像机[J].中国图象图形学报,2004(07):82-88.

[6]Zhengyou Zhang. A Flexible New Technique for Camera Calibration.[J]. IEEE Trans. Pattern Anal. Mach. Intell.,2000,22(11).

附录

$$A=\begin{pmatrix}3329.2 & 2.7 & 1524.4\\ 0 & 3323.0 & 1991.8\\ 0 & 0 & 1\end{pmatrix}$$

源代码 GitHub 地址

main.m 为主程序，调用了其余函数,实现了相机的标定功能。quickSolve.m 使用了 MATLAB 提供的部分函数，同样能够实现相机标定。

1.main.m

```
clear;
%load images and data processing
imgPath='.\images\';
squareSize=20;
imgPathList = dir(strcat(imgPath,'*.jpg'));
imgNum = length(imgPathList);

imageFileNames = {1,imgNum};
if imgNum>0
    for i = 1:imgNum
      imageFileNames{i} = sprintf('images/%d.jpg', i);
    end
end

[imagePoints,boardSize,imagesUsed] =
detectCheckerboardPoints(imageFileNames);
m=permute(imagePoints,[2 1 3]);
M = generateCheckerboardPoints(boardSize,squareSize)';

% 2.2 Homography between the model plane and its image
boardCornerNum=size(imagePoints,1);
rowAllOne=ones(1,boardCornerNum);
M=[M;rowAllOne];
```

```
for i =1:imgNum
    m(3,:,i)=rowAllOne;
end

% according to the appendix A of the paper,calculate H for
each image
L = zeros(2*boardCornerNum,9);
for i = 1:imgNum
    H(:,:,i) = solveH(m(:,:,i),M);
end

%solve V (A summarized matrix to restrict B)
V = zeros(2*imgNum,6);
for i=1:imgNum
    [V(2*i-1,:),V(2*i,:)]=solveB(H(:,:,i));
end

%solve B
[u,s,v]=svd(V);
b = v(:,end);

%solve A(intrinsic matrix)
v0    = (b(2)*b(4)-b(1)*b(5))/(b(1)*b(3)-b(2)^2);
lamda = b(6)-(b(4)^2+(b(2)*b(4)-b(1)*b(5))/(b(1)*b(3)-
b(2)^2)*(b(2)*b(4)-b(1)*b(5)))/b(1);
alpha = sqrt(lamda/b(1));
beta  = sqrt((lamda*b(1))/(b(1)*b(3)-b(2)^2));
c     = -(b(2)*alpha^2*beta)/lamda;
u0    = (c*(b(2)*v0))/alpha-(b(4)*alpha^2)/lamda;
A = [alpha c u0;0 beta v0;0 0 1];

%solve RT(extrinic matrix)
%solve Ks
D = [];
d = [];
for i = 1:imgNum
    RT = solveRT(A,H(:,:,i));
    XY = RT*M;
    UV = A*XY;
    XY=[XY(1,:)./XY(3,:); XY(2,:)./XY(3,:); XY(3,:)./XY(3,:)];
    UV=[UV(1,:)./UV(3,:); UV(2,:)./UV(3,:); UV(3,:)./UV(3,:)];
    for j = 1:boardCornerNum
        D = [D; [(UV(1,j)-u0)*( (XY(1,j))^2 + (XY(2,j))^2 ),
(UV(1,j)-u0)*( (XY(1,j))^2 + (XY(2,j))^2 )^2] ;
```

```
                [(UV(2,j)-v0)*( (XY(1,j))^2 + (XY(2,j))^2 ),
(UV(2,j)-v0)*( (XY(1,j))^2 + (XY(2,j))^2 )^2]];

        d = [d; (m(1,j,i)-UV(1,j)) ; (m(2,j,i)-UV(2,j))];
    end
end
k0 = inv(D'*D)*D'*d;
k3=0;
k4=0;
param = [alpha c u0 beta v0 k0' k3 k4 ];
options = optimset('Algorithm','levenberg-marquardt');
[x,resnorm,residual,exitflag,output] = lsqnonlin(@toMinimize2,
param ,[],[], options, m, M, H);
res = resnorm/(imgNum*boardCornerNum);
alpha = x(1);c = x(2);u0 = x(3);beta = x(4);v0 = x(5);
A3 = [alpha c u0;0 beta v0;0 0 1];
k = [x(6); x(7);x(8);x(9);];

disp("A:");
disp(A3);
disp("K:");
disp(k);
disp("res:");
disp(res);
```

2.solveH.m

```
function [H,res] = solveH(m,M)
for i = 1:size(M,2)
    L(2*i-1,:) = [M(:,i)' zeros(1,3) -m(1,i)*M(:,i)'];
    L(2*i  ,:) = [zeros(1,3) M(:,i)' -m(2,i)*M(:,i)'];
end
[~,~,V] = svd(L);
H0 = V(:,end);
H0 = H0/H0(9);
H0 = H0';
options = optimset('Algorithm','levenberg-marquardt');
[x,res] = lsqnonlin(@toMinimize, H0, [], [], options, m, M);
H = reshape(x,3,3)';
end
```

3.toMinimize.m

```
function f = toMinimize(H, m, M)
H0 = reshape(H,3,3);
H0 = H0';
m_ = H0*M;
m_ = [m_(1,:)./m_(3,:);m_(2,:)./m_(3,:);m_(3,:)./m_(3,:)];
d_m = m - m_;
f = [d_m(1,:) d_m(2,:)];
end
```

3.solveH.m

```
function [H,res] = solveH(m,M)
for i = 1:size(M,2)
    L(2*i-1,:) = [M(:,i)' zeros(1,3) -m(1,i)*M(:,i)'];
    L(2*i  ,:) = [zeros(1,3) M(:,i)' -m(2,i)*M(:,i)'];
end
[~,~,V] = svd(L);
H0 = V(:,end);
H0 = H0/H0(9);
H0 = H0';
options = optimset('Algorithm','levenberg-marquardt');
[x,res] = lsqnonlin(@toMinimize, H0, [], [], options, m, M);
H = reshape(x,3,3)';
end
```

4.solveB.m

```
function [vCol1,vCol2] = solveB(H)

v12=[H(1,1)*H(1,2) ,H(1,1)*H(2,2)+H(2,1)*H(1,2) ,H(2,1)*H(2,2)
,H(3,1)*H(1,2)+H(1,1)*H(3,2), H(3,1)*H(2,2)+H(2,1)*H(3,2),
H(3,1)*H(3,2)];
    v11=[H(1,1)^2 2*H(2,1)*H(1,1) H(2,1)^2 2*H(3,1)*H(1,1)
2*H(3,1)*H(2,1) H(3,1)^2];
    v22=[H(1,2)^2 2*H(2,2)*H(1,2) H(2,2)^2 2*H(3,2)*H(1,2)
2*H(3,2)*H(2,2) H(3,2)^2];
    vCol1=reshape(v12,1,6);
    vCol2=v11-v22;
end
```

5.solveRT.m

```
function RT = Solve_Extrinsic(A,H)
lamda = (norm(inv(A)*H(:,1))+norm(inv(A)*H(:,2)))/2;
r1 = 1/lamda*inv(A)*H(:,1);
r2 = 1/lamda*inv(A)*H(:,2);
```

```
r3 = cross(r1, r2);
R = [r1 r2 r3];
[u,s,v] = svd(R);
R = u*v';
t = 1/lamda*inv(A)*H(:,3);
RT = [R(:,1) R(:,2) t];
end
```

6.toMinimize2.m

```
function f = toMinimize2(param, m, M, H)
A = [param(1) param(2) param(3); 0 param(4) param(5); 0 0 1];
k1 = param(6);
k2 = param(7);
k3 = param(8);
k4 = param(9);
u0 = param(3);
v0 = param(5);
boardCornerNum = size(m,2);
imgNum = size(m,3);
f = [];
for i = 1:imgNum
    lamda = (norm(inv(A)*H(:,1,i))+norm(inv(A)*H(:,2,i)))/2;
    r1 = 1/lamda*inv(A)*H(:,1,i);
    r2 = 1/lamda*inv(A)*H(:,2,i);
    r3 = cross(r1, r2);
    R = [r1 r2 r3];
    [u,s,v] = svd(R);
    R = u*v';
    t = 1/lamda*inv(A)*H(:,3,i);
    RT = [R(:,1) R(:,2) t];
    XY = RT*M;
    UV = A*XY;
    XY=[XY(1,:)./XY(3,:); XY(2,:)./XY(3,:); XY(3,:)./XY(3,:)];
    UV=[UV(1,:)./UV(3,:); UV(2,:)./UV(3,:); UV(3,:)./UV(3,:)];
    for j = 1:boardCornerNum
        UV(1,j) = UV(1,j) + (UV(1,j) -
u0)*(k1*((XY(1,j))^2+(XY(2,j))^2) +
k2*((XY(1,j))^2+(XY(2,j))^2)^2+2*k3*UV(1,j)*UV(2,j)+k4*(UV(1,j
))^2+(XY(1,j))^2+(XY(2,j))^2);
        UV(2,j) = UV(2,j) + (UV(2,j) -
v0)*(k1*((XY(1,j))^2+(XY(2,j))^2) +
k2*((XY(1,j))^2+(XY(2,j))^2)^2+2*k3*UV(1,j)*UV(2,j)+k4*(UV(2,j
))^2+(XY(1,j))^2+(XY(2,j))^2);
```

```
    end
    m_ = UV;
    m_ = [m_(1,:)./m_(3,:) ; m_(2,:)./m_(3,:);
m_(3,:)./m_(3,:)];
    dm = m(:,:,i) - m_;
    f = [f dm(1,:) dm(2,:) k3 k4];
end
end
```