Metric rectification using the images of two circles

perform a metric rectification from two/three images of two circles Be aware that the image dual to the circular points is not estimated accurately, thus we can rely to a different strategy, enforcing ellipses to be circles. A useful reference: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4937883

.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.

Image Analysis and Computer Vision Politecnico di Milano

Luca Magri for comments and suggestions please send an email to luca.magri@polimi.it

.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.

Contents

load the image

```
clear;
close all;
```

```
addpath('E4_aux/');
addpath('E4_data/');
im = imread('E4_data/memphis.jpeg');
im = imresize(im,0.9);
%im = imread('E4_data/sonics.jpeg');
figure; imshow(im);
```


```
do_load_ellipses = 1;
if(do_load_ellipses)
    data = load('memphis_ellipses.mat');
    e1 = data.e1;
    e2 = data.e2;
    e3 = data.e3;
end
```

select ellipses in the image

```
figure;
imshow(im);
hold all;
if(do_load_ellipses)
```

```
    e1 = drawellipse('Center',e1.Center,'SemiAxes',e1.SemiAxes,'RotationAngle',e1.RotationAngle);
    pause
    e2 = drawellipse('Center',e2.Center,'SemiAxes',e2.SemiAxes,'RotationAngle',e2.RotationAngle);
    pause
    e3 = drawellipse('Center',e3.Center,'SemiAxes',e3.SemiAxes,'RotationAngle',e3.RotationAngle);
    pause
else
    e1 = drawellipse();
    pause
    e2 = drawellipse();
    pause
end
if(0)
    save('memphis_ellipses.mat','e1','e2','e3');
end
```

convert ellipses into conic matrices

```
par_geo = [e1.Center, e1.SemiAxes,-e1.RotationAngle]';
par_alg = conic_param_geo2alg(par_geo);
[a1, b1, c1, d1, e1, f1] = deal(par_alg(1),par_alg(2),par_alg(3),par_alg(4),par_alg(5),par_alg(6));
C1=[a1 b1/2 d1/2; b1/2 c1 e1/2; d1/2 e1/2 f1];
% sanity check
im_err = zeros(size(im,1),size(im,2));
for i = 1:size(im,1)
    for j = 1:size(im,2)
        im_err(i,j) = [j,i,1]*C1*[j;i;1];
    end
end

% visual sanity check
lambda = 0.5;
figure;
imshow(lambda*im+(1-lambda)*uint8(255.*(im_err<0)));
```

convert also e2 and e3

```
par_geo = [e2.Center, e2.SemiAxes,-e2.RotationAngle]';
par_alg = conic_param_geo2alg(par_geo);
[a2, b2, c2, d2, e2, f2] = deal(par_alg(1),par_alg(2),par_alg(3),par_alg(4),par_alg(5),par_alg(6));
C2 = [a2 b2/2 d2/2; b2/2 c2 e2/2; d2/2 e2/2 f2];
```

```
par_geo = [e3.Center, e3.SemiAxes,-e3.RotationAngle]';
par_alg = conic_param_geo2alg(par_geo);
[a3, b3, c3, d3, e3, f3] = deal(par_alg(1),par_alg(2),par_alg(3),par_alg(4),par_alg(5),par_alg(6));
C3 = [a3 b3/2 d3/2; b3/2 c3 e3/2; d3/2 e3/2 f3];

% visual check
im_err2 = zeros(size(im,1),size(im,2));
im_err3 = zeros(size(im,1),size(im,2));
for i = 1:size(im,1)
    for j = 1:size(im,2)
        im_err2(i,j) = [j,i,1]*C2*[j;i;1];
        im_err3(i,j) = [j,i,1]*C3*[j;i;1];
    end
end


msk = (im_err <0) + (im_err2<0) + (im_err3<0);
lambda = 0.5;
figure;
imshow(lambda*im + (1-lambda)*uint8(255.*msk));
```

intersect ellipses

see golub for altenative solutions

```
syms 'x';
syms 'y';

% every conic provide a 2nd degree equation
eq1 = a1*x^2 + b1*x*y + c1*y^2 + d1*x + e1*y + f1;
eq2 = a2*x^2 + b2*x*y + c2*y^2 + d2*x + e2*y + f2;
eq3 = a3*x^2 + b3*x*y + c3*y^2 + d3*x + e3*y + f3;
% solve a system with two conics: this gives a 4th degree equation
eqns = [eq1 == 0, eq2 == 0];
S12 = solve(eqns, [x,y], 'IgnoreAnalyticConstraints',true,'Maxdegree',4);
% hence you get 2 pairs of complex conjugate solution
s1 = [double(S12.x(1)); double(S12.y(1)); 1];
s2 = [double(S12.x(2)); double(S12.y(2)); 1];
s3 = [double(S12.x(3)); double(S12.y(3)); 1];
s4 = [double(S12.x(4)); double(S12.y(4)); 1];
```

disambiguate between the solutions

if you have other conics in the image you can solve similar systems

```
S13 = solve([eq1 == 0, eq3 == 0], [x,y]);
S23 = solve([eq2 == 0, eq3 == 0], [x,y]);

% and get solutions
p1 = [double(S13.x(1)); double(S13.y(1)); 1];
p2 = [double(S13.x(2)); double(S13.y(2)); 1];
p3 = [double(S13.x(3)); double(S13.y(3)); 1];
p4 = [double(S13.x(4)); double(S13.y(4)); 1];

q1 = [double(S23.x(1)); double(S23.y(1)); 1];
q2 = [double(S23.x(2)); double(S23.y(2)); 1];
q3 = [double(S23.x(3)); double(S23.y(3)); 1];
q4 = [double(S23.x(4));double(S23.y(4));1];


% there should be a pair of solution that is common represent the image of the circular
% points
[s1,s2,s3,s4; p1,p2,p3,p4;q1,q2,q3,q4]
```

```
ans =

   1.0e+03 *
```

```
   0.9579 - 1.9327i   0.9579 + 1.9327i   0.9944 + 0.0008i   0.9944 - 0.0008i
  -0.5009 + 0.0233i  -0.5009 - 0.0233i   0.2175 + 0.1557i   0.2175 - 0.1557i
   0.0010 + 0.0000i   0.0010 + 0.0000i   0.0010 + 0.0000i   0.0010 + 0.0000i
   1.2067 - 0.0238i   1.2067 + 0.0238i   0.7135 + 2.0948i   0.7135 - 2.0948i
   0.2434 - 0.0752i   0.2434 + 0.0752i  -0.5797 - 0.1011i  -0.5797 + 0.1011i
   0.0010 + 0.0000i   0.0010 + 0.0000i   0.0010 + 0.0000i   0.0010 + 0.0000i
   0.7852 + 0.0182i   0.7852 - 0.0182i   1.0471 - 1.7145i   1.0471 + 1.7145i
   0.2436 - 0.0755i   0.2436 + 0.0755i  -0.4274 - 0.0290i  -0.4274 + 0.0290i
   0.0010 + 0.0000i   0.0010 + 0.0000i   0.0010 + 0.0000i   0.0010 + 0.0000i
```

disambiguate between the solutions

let's compute the line at infinity. the line at infinity in the correct position is the one from the actual image of circular points compute the line at infinity

```
l_inf_1 = imag(cross(s1,s2));
l_inf_2 = imag(cross(s3,s4));
% also close form solutions are avilable
%l_inf_1 = [imag(s1(2)),-imag(s1(1)), imag(s1(1)*s2(2))];
%l_inf_2 = [imag(s3(2)),-imag(s3(1)), imag(s3(1)*s4(2))];

l_inf_1  = l_inf_1./norm(l_inf_1);
l_inf_2  = l_inf_2./norm(l_inf_2);

% plot the image of the line at infinity
% get the intersection with x=0 and x = size(im,1)
% ax +by +c = 0, x = 0 -> y = -c/b
%           = 0, x = w -> y = -(c+aw)/b
w = size(im,2);
A1 = [0; -l_inf_1(3)/l_inf_1(2);1];
B1 = [w; -(l_inf_1(3)+l_inf_1(1)*w)/l_inf_1(2);1];

A2 = [0; -l_inf_2(3)/l_inf_2(2);1];
B2 = [w; -(l_inf_2(3)+l_inf_2(1)*w)/l_inf_2(2);1];


MSZ = 10;
figure;

hold all;
plot(A1(1),A1(2),'ro','MarkerSize',MSZ);
plot(B1(1),B1(2),'ro','MarkerSize',MSZ);
plot(A2(1),A2(2),'bo','MarkerSize',MSZ);
plot(B2(1),B2(2),'bo','MarkerSize',MSZ);
line([A1(1),B1(1)],[A1(2),B1(2)],'linewidth',4,'Color','r');
line([A2(1),B2(2)],[A2(2),B2(2)],'linewidth',4,'Color','b');
imshow(im);
axis equal;
```

let's also check how the corresponding homographies transform the image

```
H1 = [eye(2),zeros(2,1); l_inf_1(:)'];
H2 = [eye(2),zeros(2,1); l_inf_2(:)'];

tform1 = projective2d(H1');
tform2 = projective2d(H2');
J1 = imwarp(im,tform1);
%J2 = imwarp(im,tform2);
figure;
%subplot(1,2,1);
imshow(J1);
%subplot(1,2,2);
%imshow(J2);
%axis equal;
% the correct pairs of points give rise to an homography that perform a
% correct affine reconstruction.
```

let's try to get a metric transformation

```
% compute the image of the dual conic to principal points
II = s1;%mean([s1,p1,q1],2)
JJ = s2;%mean([s2,p2,q2],2)
imDCCP = II*JJ' + JJ*II';
imDCCP = imDCCP./norm(imDCCP);

C = [eye(2),zeros(2,1);zeros(1,3)]; % canonic conic dual to circular points
```

extract the line at infinity

```
l_inf = null(imDCCP);
% s1 and s2 passes through the line at infinity
s1'*l_inf
s2'*l_inf

H = [eye(2),zeros(2,1); l_inf(:)'];
tform = projective2d(H');
im_aff = imwarp(im,tform);
figure;
imshow(im_aff);
```

```
ans =

  -2.2204e-16 - 1.1102e-16i


ans =

  -2.2204e-16 + 1.1102e-16i
```

it is not possible to compute the rectifying homography

```
[V,D] = eigs(imDCCP)
% sort the eigenvalue
[d,ind] = sort(abs(diag(D)),'descend');
Ds = D(ind,ind);
Vs = V(:,ind);

D
% has a negative eigenvalue :(
```

```
V =

    0.9905    0.1377    0.0000
    0.1377   -0.9905    0.0020
   -0.0003    0.0020    1.0000


D =

   -1.0000         0         0
         0    0.1080         0
         0         0   -0.0000


D =
```

```
   -1.0000         0         0
         0    0.1080         0
         0         0   -0.0000
```

However we have a circle

so we can distil metric information let's use the information that the axis of the ellipses should be equal transform the conic in the rectified plane according to the rule C = H^-t C H^-1

```
Q = inv(H)'*C1*inv(H);
Q = Q./Q(3,3);
% sanity check
im_err = zeros(size(im_aff,1),size(im_aff,2));
for i = 1:size(im_err,1)
    for j = 1:size(im_err,2)
        im_err(i,j) = [j,i,1]*Q*[j;i;1];
    end
end
lambda = 0.5;
figure;
imshow(lambda*im_aff+(1-lambda)*uint8(255.*(im_err<0)));
```

convert the conic coefficient to geometric parameters

Ax^2 + Bxy + Cy^2 +Dx + Ey + F = 0

```
par_geo = AtoG([Q(1,1),2*Q(1,2),Q(2,2),2*Q(1,3),2*Q(2,3),Q(3,3)]);
center = par_geo(1:2);
axes = par_geo(3:4);
angle = par_geo(5);
figure;
imshow(im_aff);
hold on;
plot(center(1),center(2),'ro','Markersize',20);
```

Now we can compute the affinity that make the axis of the ellipses to be equal.

The affinity is composed by a rotation a scaling and the inverse rotation

```
alpha = angle;
a = axes(1);
b = axes(2);
% rotation
U = [cos(alpha), -sin(alpha); sin(alpha), cos(alpha)];
% rescaling the axis to be unitary
S = diag([1, a/b]);
K = U*S*U';
A = [K zeros(2,1); zeros(1,2),1];
T = A*H; % the final transformation is the composition between the homography that maps the image of the line at infinity to its canonical positic
```

apply the transformation

```
tform = projective2d(T');
J = imwarp(im,tform);
figure;
imshow(J);
title('Metric rectification.')
```

Metric rectification.

Published with MATLAB® R2021b