Metric rectification using the images of one cir

perform a metric rectification from the image of a circle. Using the image dual to circular points leads to poor results, thus we follow a different apporach.p

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

Image Analysis and Computer Vision Politecnico di Milano

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

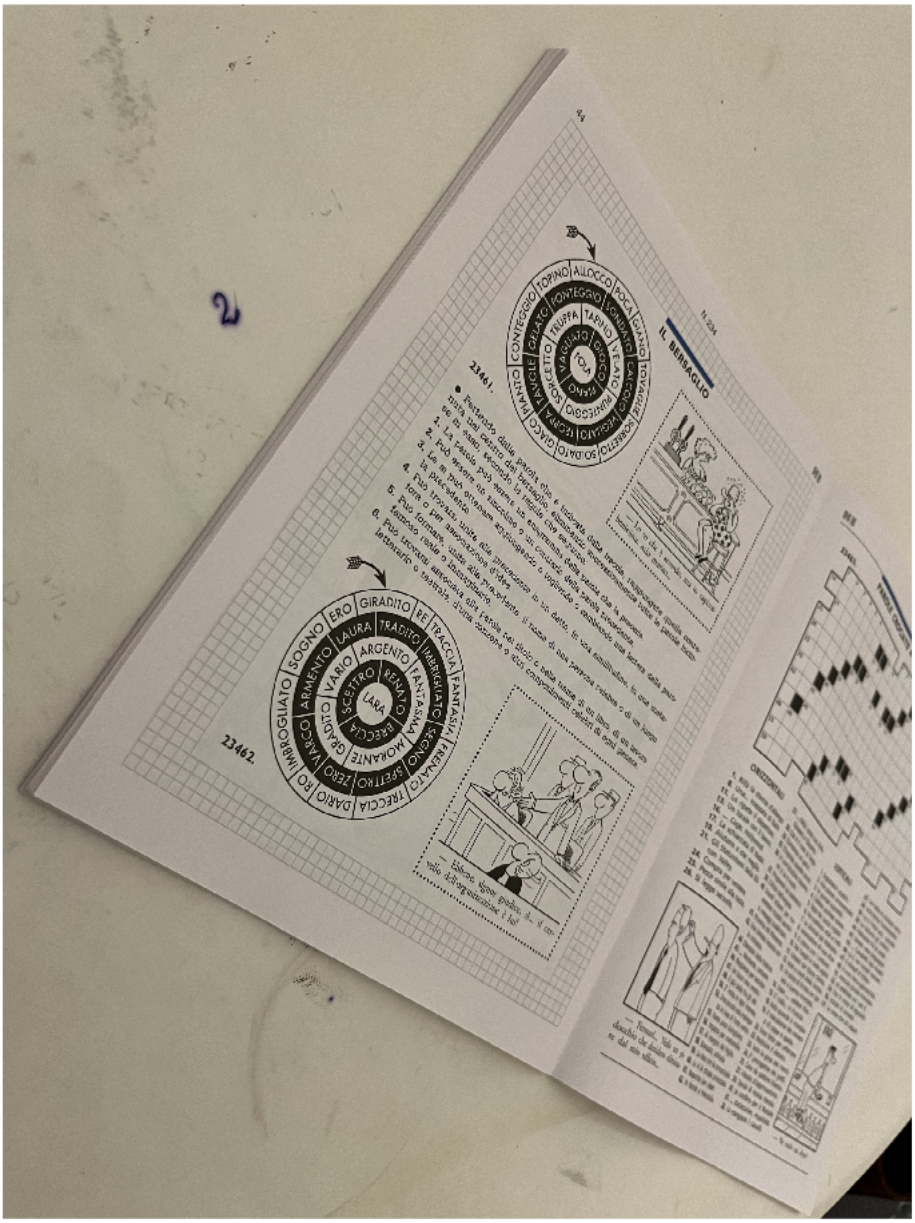
.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.\\°//.

Contents

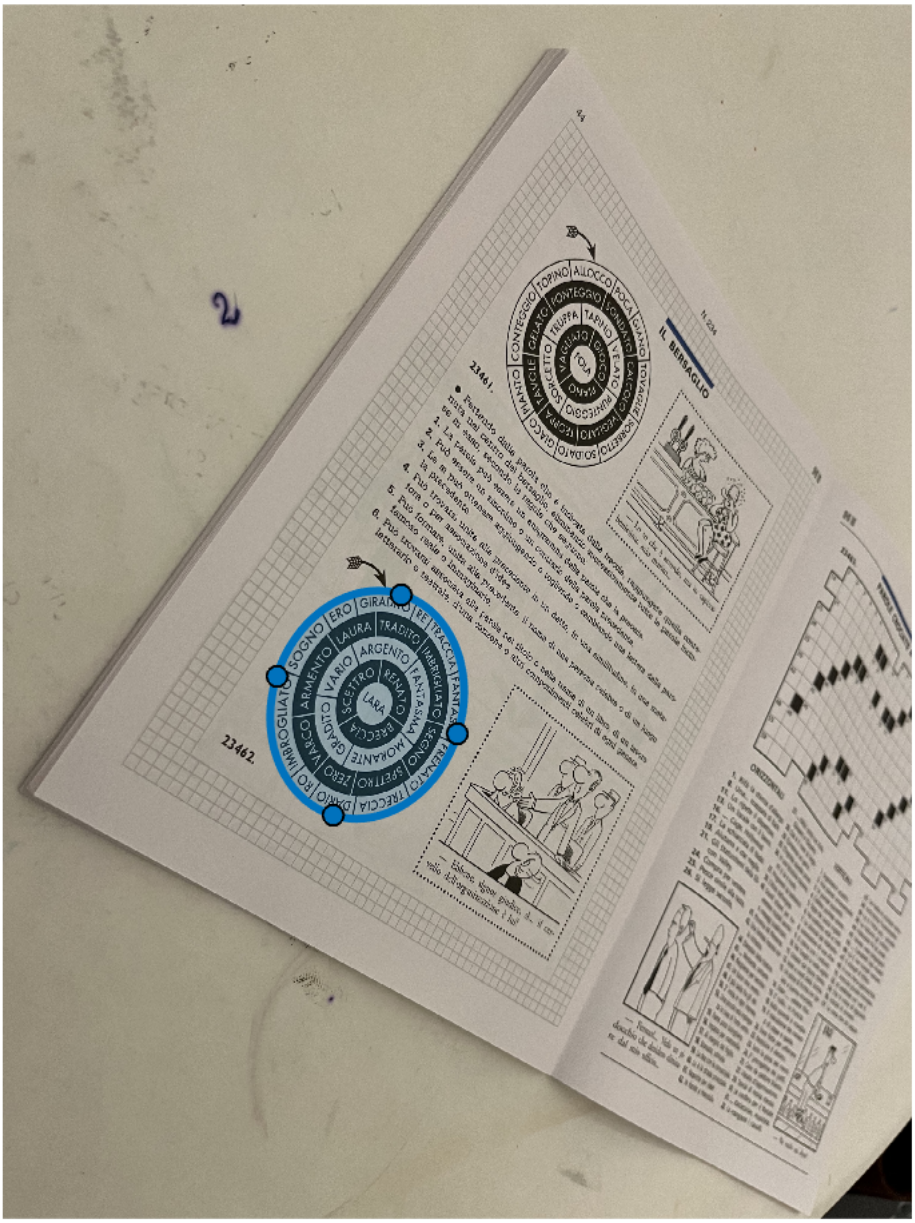
load the image

```
clear;
close all;
```

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




```
do_load_annotations = 1;
if(do_load_annotations)
    data = load('bersaglio_annotations.mat');
    ellipse1 = data.e1;
    l1 = data.l1;
    l2 = data.l2;
    m1 = data.m1;
    m2 = data.m2;
end
% select one ellipse in the image
figure;
imshow(im);
hold all;
if(do_load_annotations)
    ellipse1 = drawellipse('Center',ellipse1.Center,'SemiAxes',ellipse1.SemiAxes,'RotationAngle',ellipse1.RotationAngle);
else
    ellipse1 = drawellipse();
end
if(1)
    save('bersaglio_ellipses.mat','ellipse1');
end
```



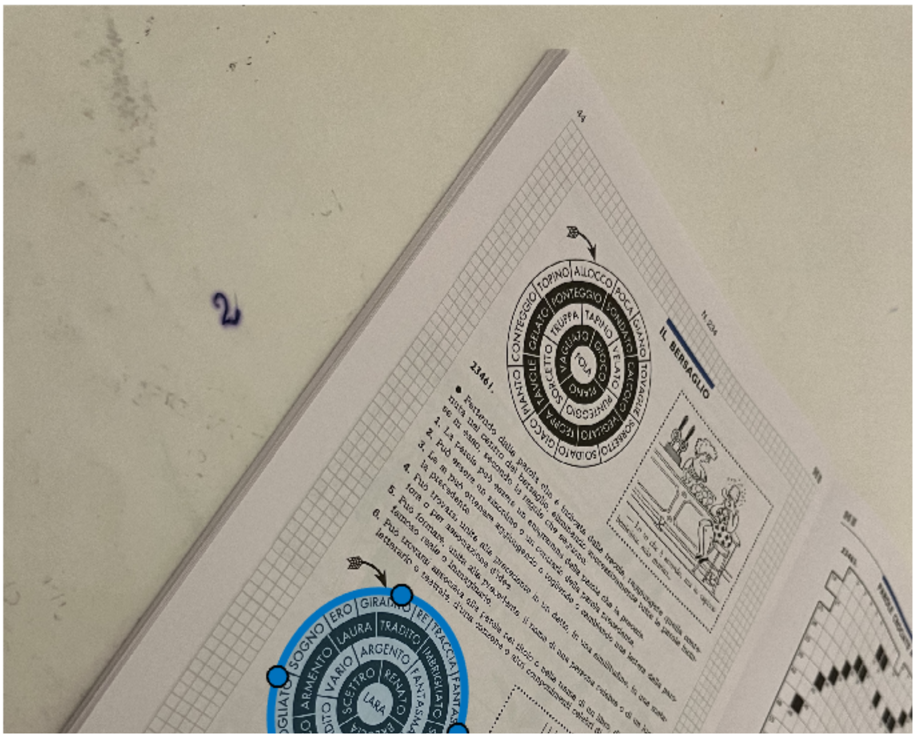
select pairs of parallel lines

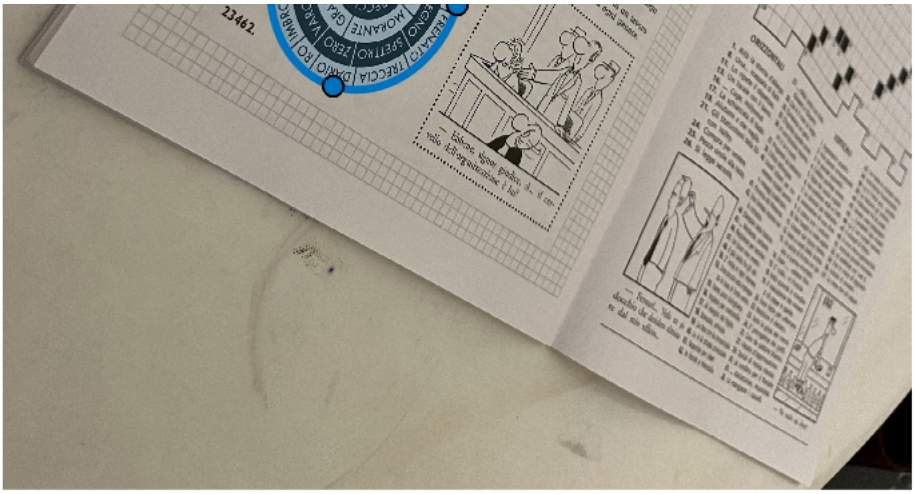
```
figure(gcf);
if(do_load_annotations==0)

    segment1 = drawline('Color','r');
    segment2 = drawline('Color','r');
    l1 = segToLine(segment1.Position);
    l2 = segToLine(segment2.Position);
    segment1 = drawline('Color','b');
    segment2 = drawline('Color','b');
    m1 = segToLine(segment1.Position);
    m2 = segToLine(segment2.Position);
end
```

compute the vanishing points

```
L = cross(l1,l2);
L = L./L(3);
M = cross(m1,m2);
M = M./M(3);

% compute the image of the line at infinity
imLinfty = cross(L,M);
imLinfty = imLinfty./(imLinfty(3));

% dispaly the selection
figure;
hold all;
% plot vanishing points
plot(L(1),L(2),'r.','MarkerSize',100);
plot(M(1),M(2),'b.','MarkerSize',100);
imshow(im);
```

```
% plot vanishing line
line([L(1),M(1)],[L(2),M(2)],'Color','Green','Linewidth',3);
```


IL BERSAGLIO
23461.



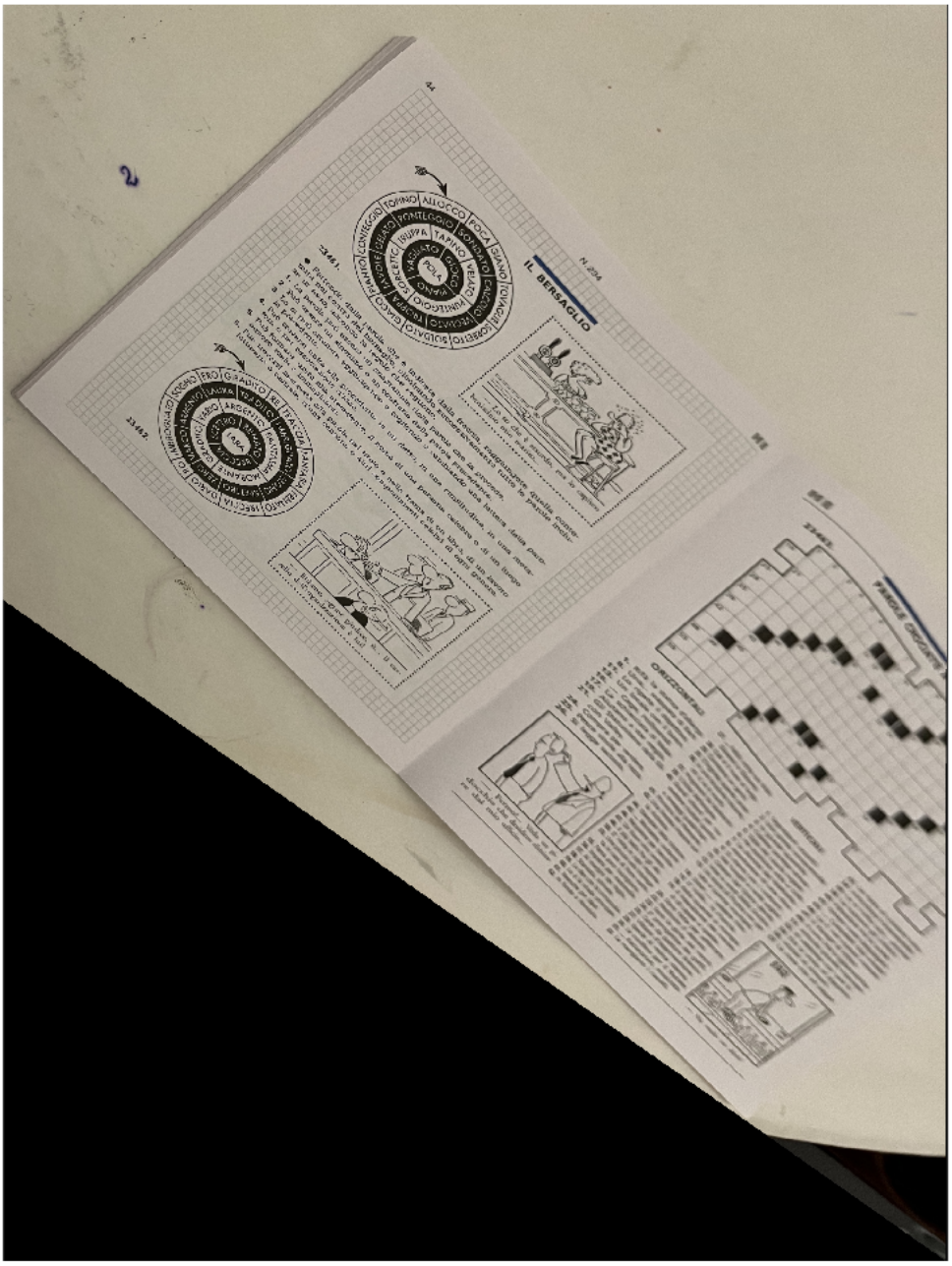
build the rectification matrix

```
H = [eye(2),zeros(2,1); imLinfty(:)'];
% we can check that H^-T* imLinfty is the line at infinity in its canonical
% form:

% compute the rectifying matrix
fprintf('The vanishing line is mapped to:\n');
disp(inv(H)'*imLinfty);
```

```
The vanishing line is mapped to:
     0
     0
     1
```

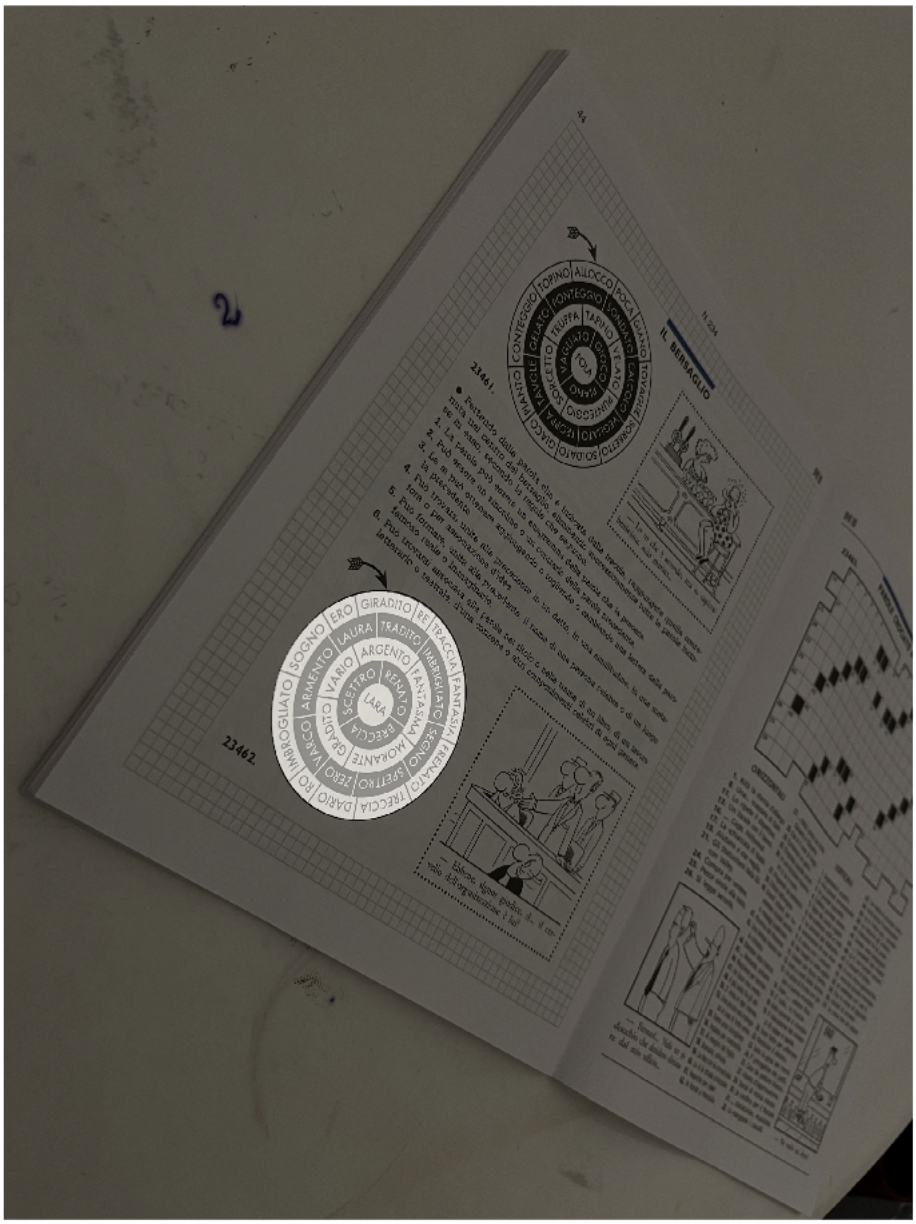
rectify the image and show the result

```
tform = projective2d(H');
J = imwarp(im,tform);

figure;
imshow(J);
imwrite(J,'E4_data/bersaglio_aff_rect.jpg');
```



convert ellipses into conic matrices

```
par_geo = [ellipse1.Center, ellipse1.SemiAxes,-ellipse1.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];
C1 = C1./C1(3,3);
% 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)));
```



intersect ellipse with the vanishing line

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 = imLinfty(1)*x + imLinfty(2)*y+ imLinfty(3);
% solve a system with a line and the conic: this gives a 2th 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];
```

let's check the solution using a direct substitution

express x = alpha y + beta

```
alpha = -imLinfty(2)/imLinfty(1);
beta = -imLinfty(3)/imLinfty(1);
% substitute in eq2 to get a second degree eq in y
% a1*(alpha y + beta)^2 + b1*(alpha y + beta)*y + c1*y^2 + d1*(alpha y + beta) + e1*y + f1
poli = [a1*alpha*alpha + b1*alpha + c1,... % coefficients of y^2
    2*a1*alpha*beta + b1*beta + d1*alpha + e1,...
    a1*beta*beta + d1*beta + f1];% coefficients of y^2

sol = roots(poli);
y1 = sol(1);
y2 = sol(2);
x1 = alpha*y1 + beta;
```

```
x2 = alpha*y2 + beta;

q1 = [x1;y1;1];
q2 = [x2;y2;1];

disp([s1 s2])
disp([q1 q2])
```

```
   1.0e+03 *

   5.1707 - 0.0074i   5.1707 + 0.0074i
   1.5503 - 4.7445i   1.5503 + 4.7445i
   0.0010 + 0.0000i   0.0010 + 0.0000i

   1.0e+03 *

   5.1707 + 0.0074i   5.1707 - 0.0074i
   1.5503 + 4.7445i   1.5503 - 4.7445i
   0.0010 + 0.0000i   0.0010 + 0.0000i
```

image of the circular points

```
II = s1;
JJ = s2;
```

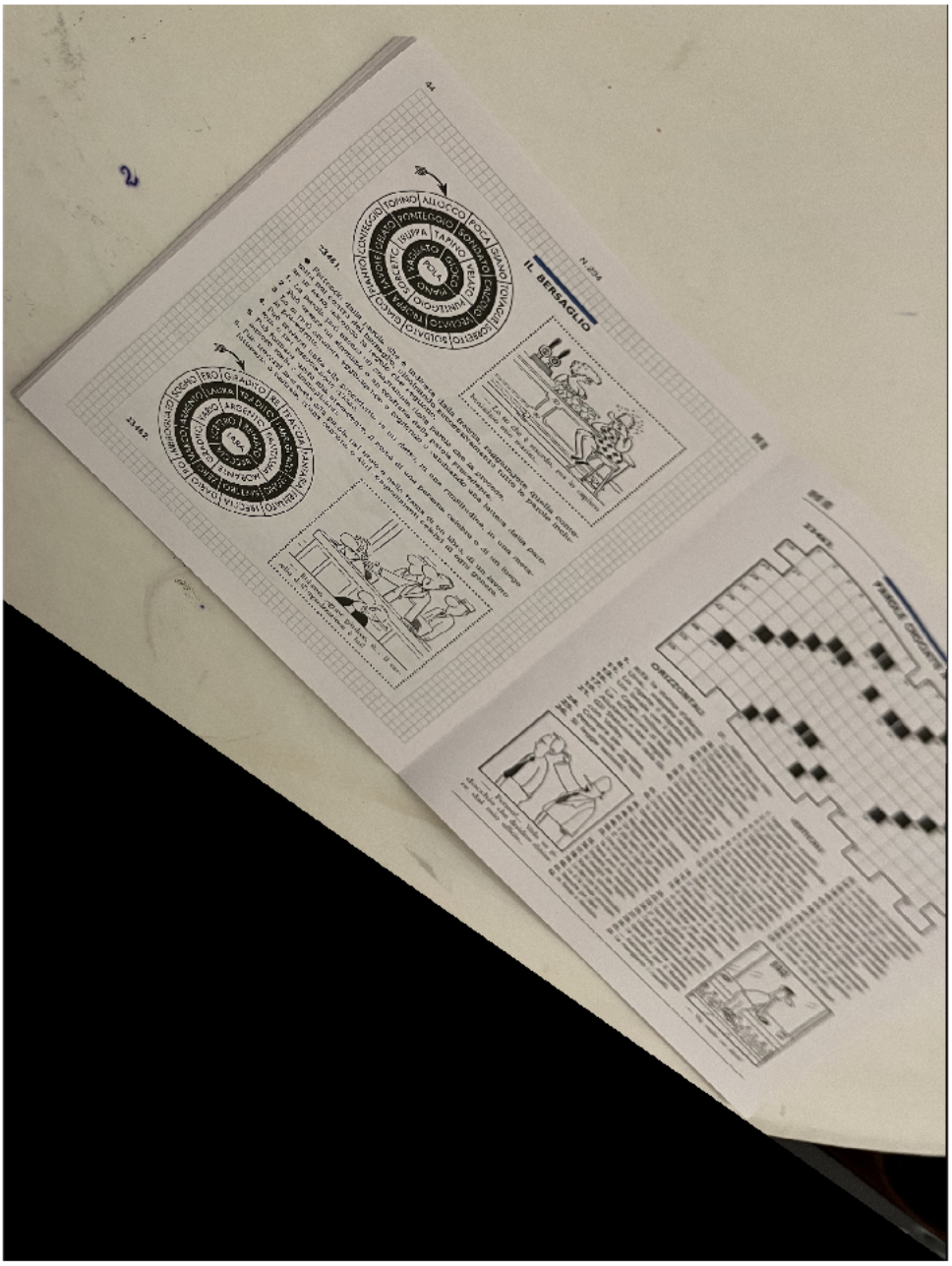
Conic dual to circular points

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

imDCCP = II*JJ' + JJ*II';
%imDCCP = imDCCP./norm(imDCCP);
```

extract the line at infinity

```
l_inf = null(imDCCP);
% II and JJ passes through the line at infinity by construction
II'*l_inf
JJ'*l_inf

% they actually coincide with the solution retrieved before
l_inf - imLinfty


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

```
ans =

  -6.1800e-18 - 2.6455e-17i


ans =

  -6.1800e-18 + 2.6455e-17i


ans =

   1.0e-07 *

    0.0000
   -0.0000
   -0.1872
```



compute angles using imDCCP

figure; imshow(im); segment1 = drawline('Color','r'); segment2 = drawline('Color','r'); l = segToLine(segment1.Position); m = segToLine(segment2.Position);

since the conic has one negative eigenvalue this doesn't work! cos_theta = (l'*imDCCP*m)/(sqrt(l'*imDCCP*l)*sqrt(m'*imDCCP*m));

compute the rectifying homography

```
%[U,D,V] = svd(imDCCP);
[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!!! We cannot use it to extract the rectifying
% homography :(
```

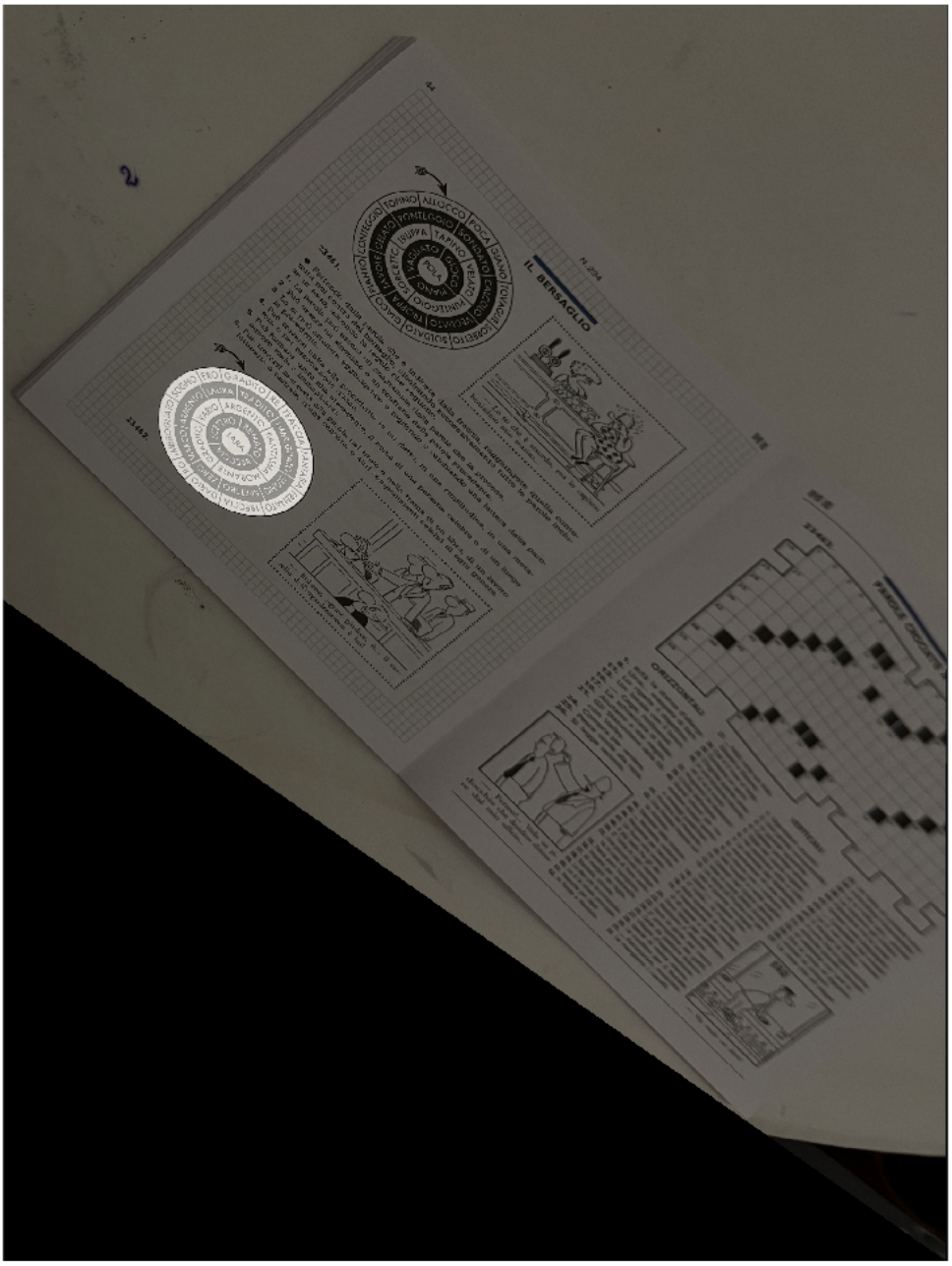
```
V =

   -0.9865   -0.1635   -0.0002
   -0.1635    0.9865    0.0000
   -0.0002   -0.0000    1.0000


D =

   1.0e+07 *
```

```
    5.6117         0         0
         0   -4.2858         0
         0         0   -0.0000


D =

   1.0e+07 *

    5.6117         0         0
         0   -4.2858         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

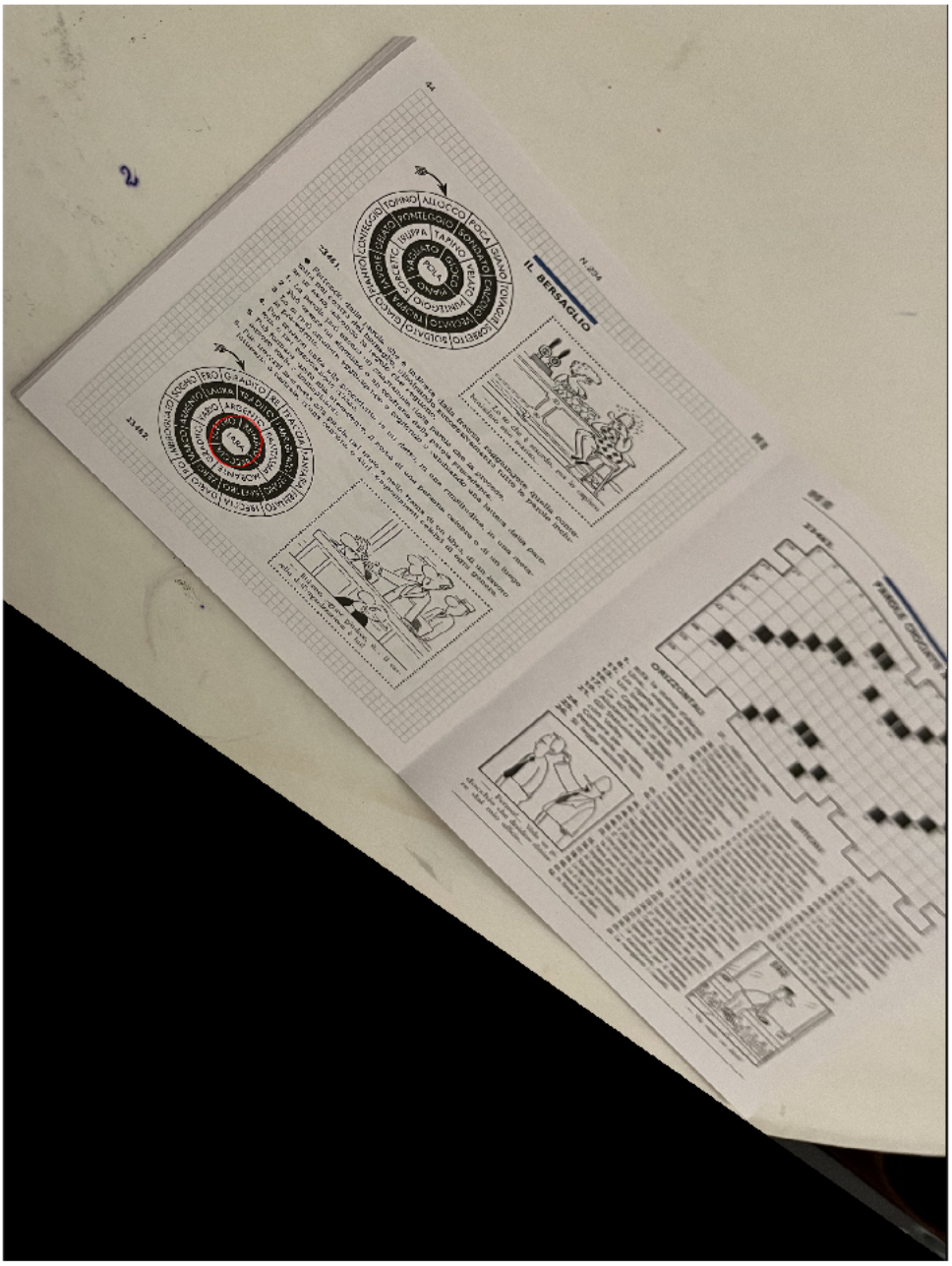
```
Q = inv(H)'*C1*inv(H);
Q = Q./Q(3,3);
% sanity check
im_aff = J;
im_err = zeros(size(im_aff,1),size(im_aff,2));
for i = 1:size(im,1)
    for j = 1:size(im,2)
        im_err(i,j) = [j,i,1]*Q*[j;i;1];
    end
end
lambda = 0.5;
figure;
imshow(lambda*im_aff+(1-lambda)*uint8(255.*(im_err<0)));
```



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.

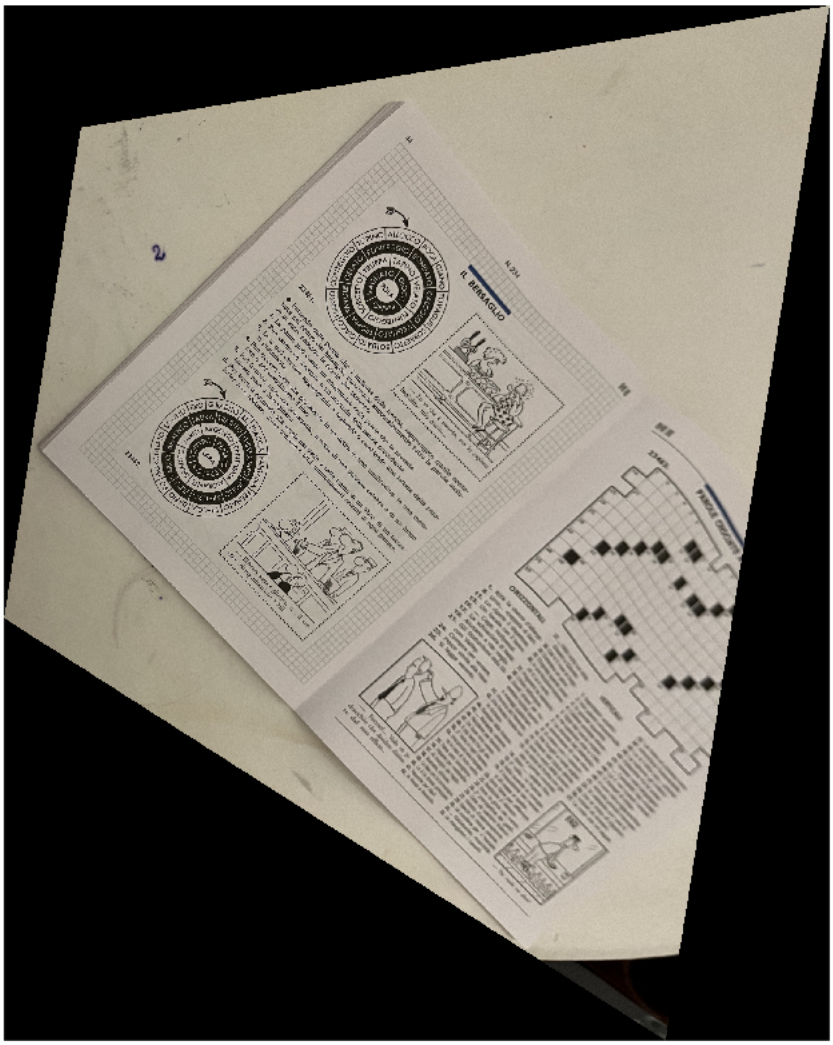
The affinity is composed by a rotation a scaling and the inverse rotation

```
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