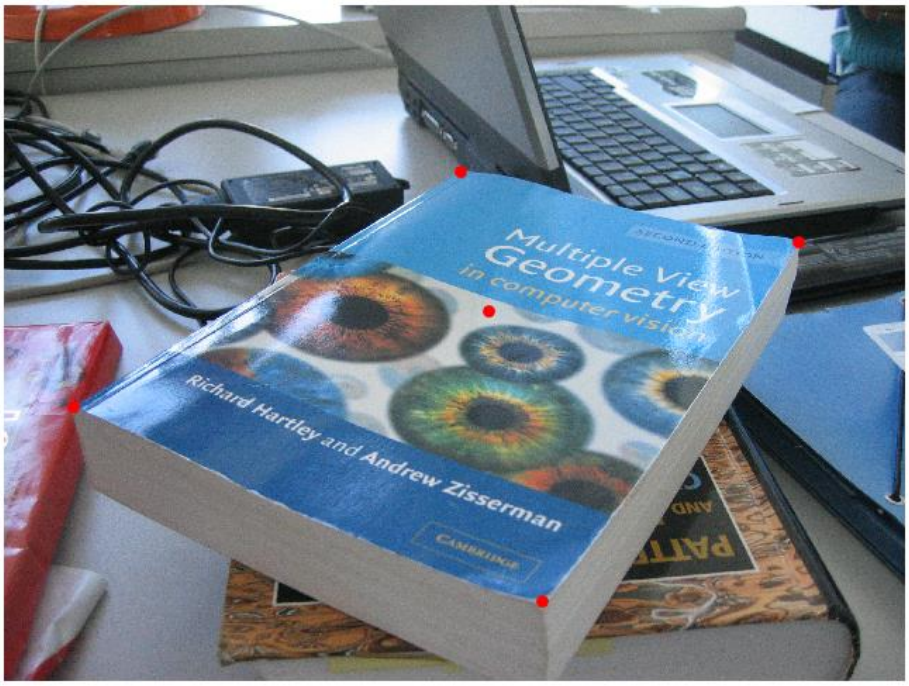
Pose from a planear target - Stéphane Maillot

Homography computation

```
im = imread('mvg.bmp');
corners_pic = [ 345, 218 ; 50, 286 ; 383, 424 ; 565, 169 ; 325, 119 ];
l = 17.3;
L = 24.6;
corners_real = [ 0, 0 ; -l/2, -L/2 ; l/2, -L/2 ; l/2, L/2 ; -l/2, L/2 ];
imshow(im)
hold on;
plot(corners_pic(:,1), corners_pic(:,2), '.r', 'MarkerSize', 15)

[U, S, V] = svd(DLT_system(corners_pic, corners_real));
S
```

```
S =

   1.0e+04 *

  Columns 1 through 9

    1.1529         0         0         0         0         0         0         0         0
         0    0.8120         0         0         0         0         0         0         0
         0         0    0.0886         0         0         0         0         0         0
         0         0         0    0.0023         0         0         0         0         0
         0         0         0         0    0.0017         0         0         0         0
         0         0         0         0         0    0.0012         0         0         0
         0         0         0         0         0         0    0.0006         0         0
         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0    0.0002         0
         0         0         0         0         0         0         0         0    0.0000
```

The last column of V will then be the kernel of this matrix.




```
ker = V(:,end);
H = reshape(ker, 3, 3)'
```

```
H =

    0.0295    0.0393    0.8430
    0.0047   -0.0084    0.5356
   -0.0000    0.0000    0.0024
```

If we use -H, the Z-axis will be reversed.

Pose estimation

```
A = [ 800, 0, 320 ; 0, 800, 240 ; 0, 0, 1 ]
G = A^-1 * H;
R1 = G(:,1);
R2 = G(:,2);
R3 = cross(R1, R2);
t = G(:,3);
```

```
A =

   800     0   320
     0   800   240
     0     0     1
```

Normalization

```
l = sqrt(norm(R1) * norm(R2));
R1 = R1 / l;
R2 = R2 / l;
t = t / l
c = R1 + R2;
d = cross(c, R3);
R1 = 1 / sqrt(2) * (c / norm(c) + d / norm(d));
R2 = 1 / sqrt(2) * (c / norm(c) - d / norm(d));
R3 = cross(R1, R2);
R = [R1, R2, R3]
```

```
t =

    1.2424
   -1.0812
   40.7744
```

```
R =

    0.8770    0.4804   -0.0048
    0.2156   -0.4025   -0.8897
   -0.4294    0.7792   -0.4566
```

As R and t are the rotation and translation to convert a point in the book coordinates into camera coordinates.
C = -R't is the coordinate of the camera in the book frame.

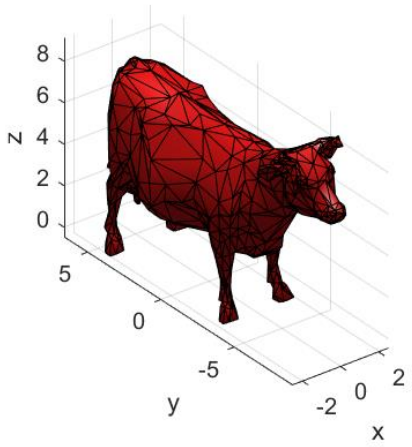
```
C = -R'*t
P = A * [R, t]
```

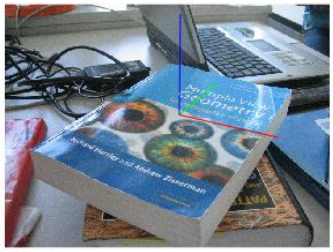
```
C =

   16.6501
  -32.8042
   17.6612
```

```
P =

   1.0e+04 *

    0.0564    0.0634   -0.0150    1.4042
    0.0069   -0.0135   -0.0821    0.8921
   -0.0000    0.0001   -0.0000    0.0041
```

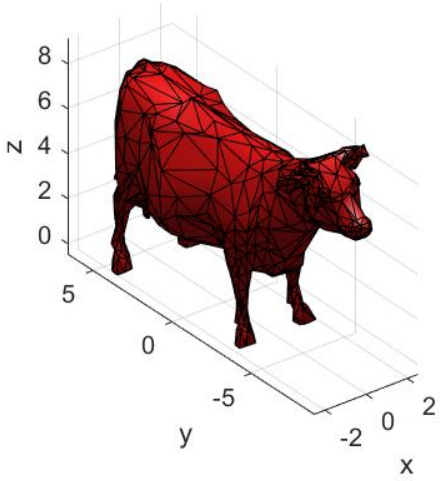
Projection of a virtual object

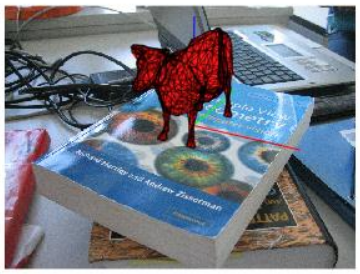
```
load('model3d-cow.mat','fv');
V=fv.vertices;
F=fv.faces;
nV = size(V, 1);

%rotate and translate model 3d to align it with the book
V = V * rotx(-90);
V = V * rotz(90);
V = V + [0, 0, 5];

%plot original model in 3D
figure; subplot(1,2,1);
trisurf(F,V(:,1),V(:,2),V(:,3),'FaceColor',[1,0.1,0.1],...
'EdgeColor',[0.0 0.0 0.0]);
light('Position',[-1.0,-1.0,100.0],'Style','infinite');
lighting phong;
xlabel('x');
ylabel('y');
zlabel('z');
axis equal;


%project model onto the image
frame = [ 0, 0, 0 ; 10, 0, 0 ; 0, 10, 0 ; 0, 0, 10];

frame = (P * [frame, ones(4, 1)]')';
frame = frame(:,1:2) ./ frame(:,end);

proj = (P * [V, ones(nV, 1)]')';
proj = proj(:,1:2) ./ proj(:,end);
uCow = proj(:,1);
vCow = proj(:,2);

%plot projected model in 2D
subplot(1,2,2); imshow(im);
line([frame(1,1), frame(2,1)], [frame(1,2), frame(2,2)], 'Color', 'r')
line([frame(1,1), frame(3,1)], [frame(1,2), frame(3,2)], 'Color', 'g')
line([frame(1,1), frame(4,1)], [frame(1,2), frame(4,2)], 'Color', 'b')
```





```
Color=repmat([1,0.1,0.1],nV,1);
fv2d.vertices=[uCow,vCow];
fv2d.faces=F;
fv2d.facevertexcdata=Color;
p = patch(fv2d,'FaceAlpha',1,'EdgeAlpha',0.25);
shading faceted;
```





Published with MATLAB® R2017a