3D Pose Estimation

C. Emiliano Solórzano Espíndola, carlosemiliano04@gmail.com

ETSEIB- Universitat Politècnica de Catalunva

***Abstract*— The present works performs a simple pose estimation using the DLT Algorithm in order to obtain the parameters matrix that can helps to translate from one coordinate system to the other.**

A. Initialization

In this part the fundamental matrices are declared, those are: the calibration matrix, that gives the parameters of the camera, and the 3D points matrix, that gives the corresponding values for the points in centimeters.

```
clc, clear all, close all

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

%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;


I=double(imread('mvg.bmp'));

%Real dimensions of the book
x0 = 17.3/2;
y0 = 24.6/2;
z0 = x0/2;

%Calibration matrix
 A = [800 ,0 ,320 ;0 ,800 ,240 ;0 ,0 ,1];

% 3D points
X1 = [-x0, y0];
X2 = [x0, y0];
X3 = [x0, -y0];
X4 = [-x0, -y0];
X = [X1; X2; X3; X4];
X(:,3) = 1;
```

B. 2D Points

In this part we can manually select those points in the image that correspond with the ones declared in the 3D points matrix. This will give us the values needed for solving the system.

```
% figure, imshow(uint8(I));
% [u, v, p] = impixel(I);   Used to get the picture coordinates the first
% time
```

```
v = [119, 171, 420, 287];
u = [328 ,562 ,379 ,53];
U = [u ;v ;zeros(1,4)];
```

C. Find the H matrix

An M matrix is declared with the system of points and since it's not a square matrix, a singular value decomposition is performed in order to obtain the eigenvectors.

```
M = [];
for i = 1:length(u)
    Mi = [X(i,:), 0,0,0, -X(i,:)*u(i);
        0,0,0, X(i,:), -X(i,:)*v(i)];
    M = [M; Mi];
end


[U1, S, V2] = svd(M);

% Pick the last column of v and reshape it into a 3x3 matrix
H = reshape(V2(:,9),3,3)'
```

```
H =

    0.0296    0.0386    0.8418
    0.0054   -0.0086    0.5375
   -0.0000    0.0000    0.0024
```

D. Find the rotation matrix

From the H matrix, the rotations and translation values can be obtained. Also some operations are performed in order to assure that the system is orthonormal.

```
G = A\H;
%Normalization
l = sqrt(norm(G(:, 1))*norm(G(:, 2)));
G = G/l;
c = G(:,1) + G(:, 2);
p = cross(G(:, 1), G(:, 2));
d = cross(c, p);
```

```
R1 = ((c/norm(c)) + (d/norm(d))) / sqrt(2);
R2 = ((c/norm(c)) - (d/norm(d))) / sqrt(2);

Rt = zeros(3, 4);
Rt(:, 1) = R1;
Rt(:, 2) = R2;
Rt(:, 4) = G(:, 3);
Rt(:, 3) = cross(R1, R2);

Rt
% Camera
C = -Rt(1:3, 1:3)'*Rt(:, 4)
```

```
Rt =

    0.8876    0.4603    0.0147    1.3244
    0.2299   -0.4152   -0.8802   -1.0302
   -0.3991    0.7846   -0.4744   42.0737


C =

   15.8524
  -34.0505
   19.0334
```

E. Projection matrix

Finally the projection matrix P is obtained with the coordination and the rotation/translation matrices. This last matrix is the one needed for passing form one system to the other in a form $U = P * X$.

```
Rcowtobook = [0 0 1; 1 0 0; 0 1 0] * inv([-1 0 0; 0 1 0; 0 0 -1]);
V = Rcowtobook*V';
V = V';

P = A * Rt;
V(:,4) = 1;
V1 = P * V';

V1(1, :) = V1(1, :) ./ V1(3, :);
V1(2, :) = V1(2, :) ./ V1(3, :);

P
```

```
P =

   1.0e+04 *

    0.0582    0.0619   -0.0140    1.4523
```

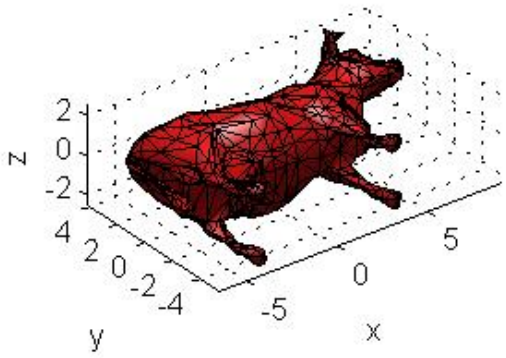
```
    0.0088   -0.0144   -0.0818    0.9274
   -0.0000    0.0001   -0.0000    0.0042
```

F. Plot projected model in 2D

In this part each point of a 3D mesh is passed from its 3D representation to the 2D defined with the new system.

```
subplot(1, 2, 2); imshow(uint8(I)), hold on;

Color=repmat([1,0.1,0.1],length(V1),1);
fv2d.vertices = [V1(1, :)', V1(2,: )'];
fv2d.faces = F;
fv2d.facevertexcdata=Color;
% p = patch(fv,'FaceAlpha',1,'EdgeAlpha',0.25);
p = patch(fv2d, 'FaceAlpha', 1, 'EdgeAlpha', 0.25), hold off;
shading faceted;
```



G. Plot the axes and a figure

In this part the axes are shown with the X in red, the Y in blue and the Z in green growing from the origin or (0, 0, 0) point in the 3D representation, each one would represent 10 centimeters in the real world. Also a simple figure is on top over the x,y plain.

```
figure, imshow(uint8(I)), hold on;
a=[0:0.2:10; zeros(1,51); zeros(1,51); ones(1,51)];
b=[zeros(1,51); 0:0.2:10; zeros(1,51); ones(1,51)];
l=[zeros(1,51); zeros(1,51); 0:0.2:10; ones(1,51)];
a1= proj(a,P);
b1= proj(b,P);
l1= proj(l,P);

plot(a1(1,:),a1(2,:),'r-','LineWidth',4)
plot(b1(1,:),b1(2,:),'b-','LineWidth',4)
plot(l1(1,:),l1(2,:),'g-','LineWidth',4)

vert = 3*[-1 -1 0;1 -1 0;1 1 0;-1 1 0;-1 -1 2;1 -1 2;1 1 2;-1 1 2];
fac = [1 2 6 5;2 3 7 6;3 4 8 7;4 1 5 8;1 2 3 4;5 6 7 8];

vert(:, 4) = 1;
vert1 = proj(vert', P);

cu.vertices = [vert1(1,:)', vert1(2,:)'];
cu.faces = fac;
Color1 = repmat([0.1,0.1,1],length(vert1),1);
cu.facevertexcdata = Color1;
patch(cu, 'FaceAlpha', .24, 'EdgeAlpha', 1)
shading faceted;
```