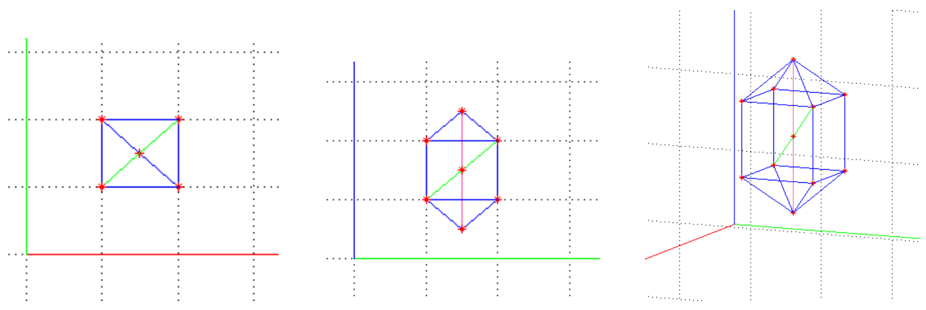
Projective Reconstruction and Calibrated SFM
Visual Perception and Structure from Motion
Université de Bourgogne

Sandeep Manandhar

April 27, 2015

1 Projective Reconstruction

We define a simple 3D scene that is sufficient in itself to observe the ***Collinearity*** and ***Coplanarity*** of the geometry.



(a) top view (b) side view (c) perspective view

Figure 1: 3D scene

1.1 Camera Parameters

Let $\mathbf{X}$ be a 3D point seen from world coordinate frame. Let p_{cam} be the same point from the camera coordinate frame. Let the camera be at $\mathbf{C}$ in the world frame transformed by $\mathbf{R}$, and $\mathbf{t}$(rotation and translation).

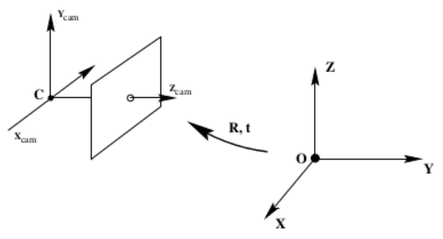


Figure 2: world to camera coordinate frame, referred from [2]

$$p_{cam} = \mathbf{R}(\mathbf{X} - \mathbf{C}) \tag{1}$$

$$p_{cam} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2}$$

This equation brings the point **X** from world coordinate to the camera frame at p_{cam}. Let **x** be the 2D image point in the camera existing as a homogeneous coordinate. Then we need a camera projection matrix to do such transformation. We define it as,

$$\mathbf{x} = \mathbf{K} * [\mathbf{I}|0] * p_{cam} \tag{3}$$

Where, **K** provides the camera's intrinsic parameters. From equation 2 and 3, we can say,

$$\mathbf{x} = \mathbf{K} * \mathbf{R} * [\mathbf{I}| - C] * \mathbf{X} \tag{4}$$

So the final transformation or the camera matrix is given as,

$$\mathbf{T} = \mathbf{K} * \mathbf{R} * [\mathbf{I}| - C] \tag{5}$$

In order to keep things simple, the stereo camera pairs have been aligned such that their axes are parallel to the world frame(i.e. no rotation) and translated along one axis only.

```
f = 5; %focal length
cx = 0; cy = 0; %image center
cam1_position = [16, 16, -35];  %%viewing up
cam2_position = [14, 16, -35];  %%viewing up
%%[R|t] matrix for the 1st camera
R1 =[...
     1     0     0   -16;
     0     1     0   -16;
     0     0     1    35;
     0     0     0     1];
%%camera intrinsics
K1 =[...
     5     0     0;
     0     5     0;
     0     0     1];
P = eye(3,3);
P =[P zeros(3,1)];
%%combining intrinsic and extrinsic parameters
T1 = K1*P*R1;
 %%T1
 T1 = [...
     5     0     0   -80;
     0     5     0   -80;
     0     0     1    35];
```

Using these camera matrices **T1** and **T2**, we project the 3D points to the image planes. We can notice **??** all of the 3D points have been projected to the image planes and have a slight disparity among them.

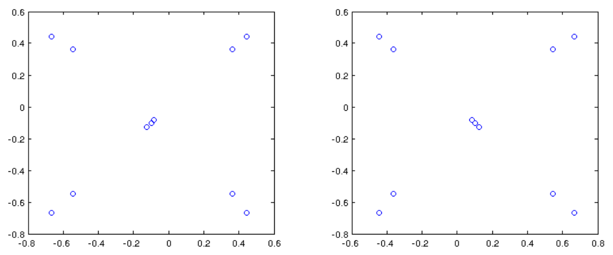


Figure 3: Projections in the left and right cameras

2 Fundamental matrix

We use these projections, assuming that we know the point correspondences to calculate the fundamental matrix between these two camera views. Since, there are 11 points in our scene, we can do so by using singular value decomposition.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%With point correspondences
%%xT*F*x = 0
%%
A = theAMatrix(x1', x2');
[au, as,av] = svd(A);
F_1 = av(:,end);
F_1 = reshape(F_1, [3,3])';
%%%%%
%%F_1 results as
F_1 =[...]
    0.0163    0.0163    0.0000;
   -0.0163   -0.0163    0.7067;
    0.0000   -0.7067   -0.0000];
```

As we can see here 4, the epipolar lines are not being drawn as expected. Since, the cameras were at pure translation only, these lines should have been horizontal.

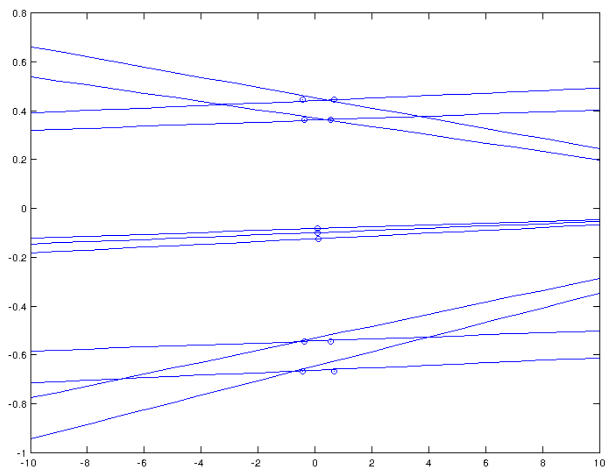


Figure 4: Drawing epipolar lines using the calculated fundamental matrix

3 Canonic Cameras

Using the canonical representation for the cameras, we set

$$\mathbf{T} = [\mathbf{I}|0] \tag{6}$$

$$\mathbf{T} = [[e']_x\mathbf{F}|e'] \tag{7}$$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%first camera matrix
P1 = [...
    1 0 0 0;
    0 1 0 0;
    0 0 1 0;
    ];
%second camera matrix
[u s v] = svd(F_1');
e = v(:,3)  %epipole e'
ex = [0 -e(3) e(2);
    e(3) 0 -e(1);
    -e(2) e(1) 0];
M = 1/norm(e) * ex*F_1;

P2 = [M e];
```

Since, we have two views defined by $P1$ and $P2$, we can solve for the 3D point X.

$$\mathbf{x} = P1 * X([p_x]P1) * X = 0Q * X = 0 \tag{8}$$

We have three unknowns for each points but each view provides, two equations. Using both of the views, we solve to get the X.

```
p3D = []; %%3D points here
for i=1:length(x1)
u1 = x1(1,i); v1 = x1(2,i); %image in 1st camera
u2 = x2(1,i); v2 = x2(2,i); %image in 2nd camera

Q = [...
    u1*P1(3,:) - P1(1,:);
    v1*P1(3,:) - P1(2,:);
    u2*P2(3,:) - P2(1,:);
    v2*P2(3,:) - P2(2,:);];

[U S V] = svd(Q);
px = V(:,end);
px = px./px(4);
p3D = [p3D ;px'];
end
```

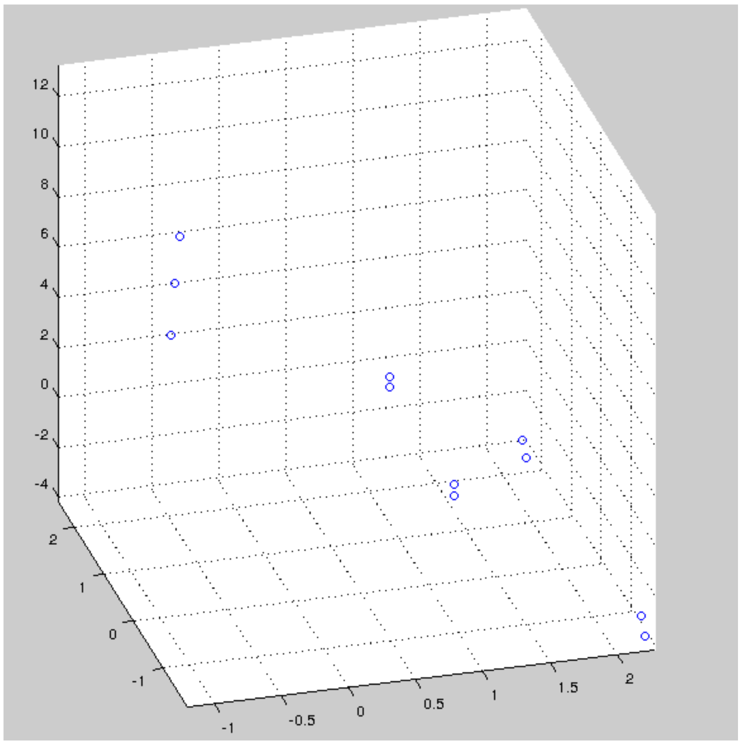


Figure 5: Reconstruction up to Projective space

Here 5, the structure looks nothing like the ground truth but the fact that coplanarity and collinearity is preserved in projective space remains true. The camera matrix $P2$ does not necessarily have orthogonal vectors. It could possibly create an affine transformation or projective. So we cannot expect metric reconstruction here.

3.0.1 Residual error

We project the found non-metric reconstructed 3D points to the camera planes and compare them with the calibrated cameras' image planes to get the resiudal error.

```
%%
plot(x1(1,:), x1(2,:), 'ob');

px1 = world2image(P1,p3D);

px2 = world2image(P2, p3D);

plot(px1(1,:), px1(2,:), '*r');
hold on;
plot(x1(1,:), x1(2,:), 'ob');

d1=norm(px1 - x1);
d2=norm(px2 - x2);
%%residual errors
d1 =     0.0079;
d3 =     0.0157;
```

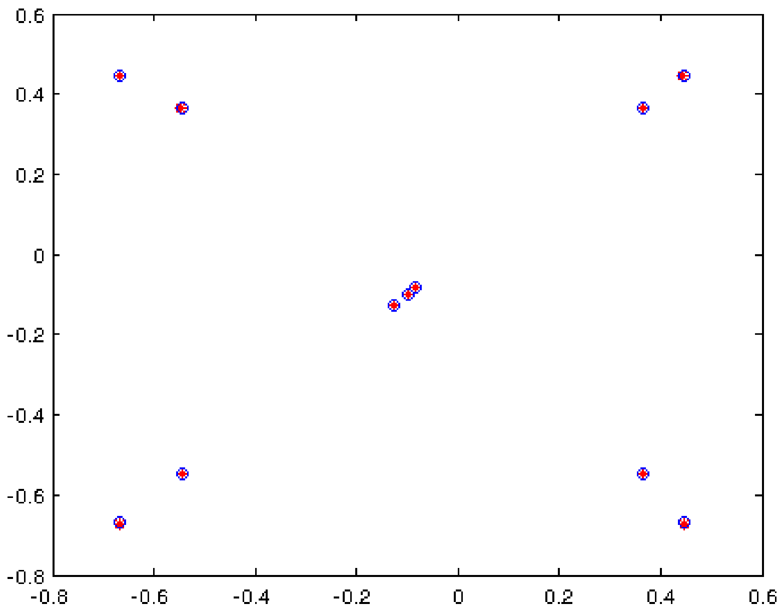


Figure 6: Projection to image planes, ground truth in blue circles, projections from canonical in red star

4 Essential matrix

We calculate the essential matrix using the calibrated cameras. We will use the camera matrices from the section 1.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%With intrinsics
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
PP = inv(R1)*R2
R = PP(1:3,1:3)
t = PP(:,end)
tx = [0 -t(3) t(2);
    t(3) 0 -t(1);
    -t(2) t(1) 0];
E = tx*R; %essential matrix

F = inv(K1)'*E*inv(K1); %%fundamental matrix
%%
%E results to
E =[...
     0     0     0;
     0     0    -2;
     0     2     0;
     ]
%%
%F results to
F =[...
    0         0         0;
    0         0   -0.4000;
    0    0.4000         0];
```

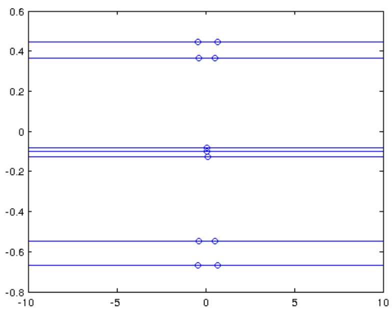


Figure 7: Epipolar line using Fundamental matrix fro E

4.1 Extract R and t from E

Following the implementation of [1], we use singular value decomposition of E matrix to get R and t. We obtain 4 solutions out of which one is the true one. The four solutions might lead to cameras rotated in one direction or the opposite and each of them might have translation in different direction along the same axis. To find the true one, we use the cheirality constraint where we reconstruct one of the point visible in each cameras and check for a positive depth value. In our case, two of the solutions gave positive depth values and since our cameras had only translation between them, we could consider these two solutions valid stereo pairs.

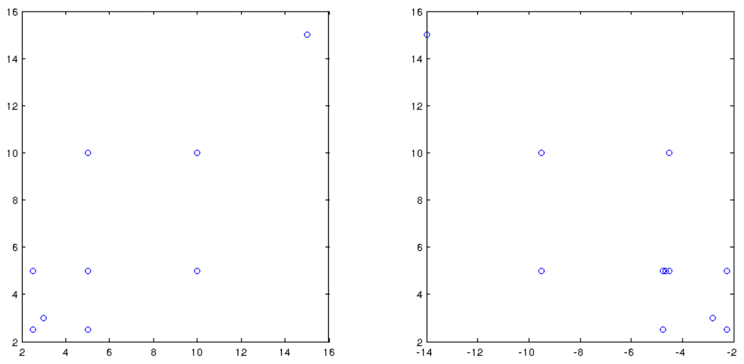


Figure 8: Projection to image planes, using P=[I—0] and one of the P' that delivers cherality constraint

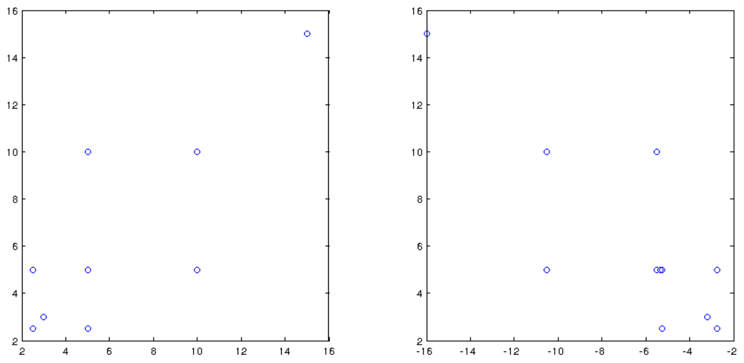
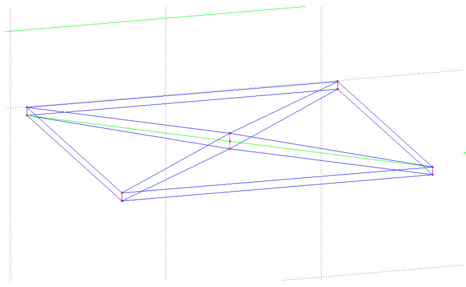


Figure 9: Projection to image planes, using P=[I—0] and one of the P' that delivers cherality constraint but translated in one axis

4.2 3D reconstruction

As we can see ,the object has been reconstructed up to similarity transformation with loss of scale. Comparing with the ground truth, we find the z-axis has been


(a) reconstructed

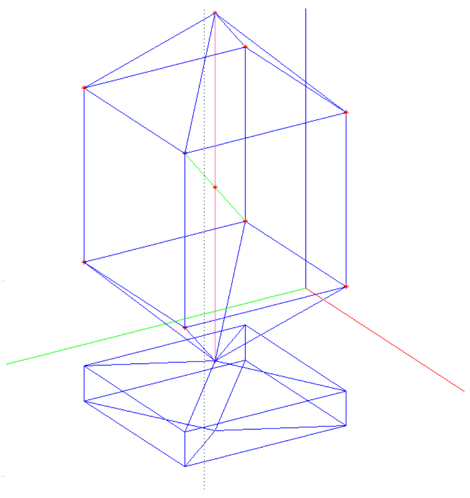

(b) comparing with original

Figure 10: 3D reconstruction

scaled down by 5.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Reconstructed 3D points
cp3D =
   10.0000   10.0000    2.0000    1.0000
   20.0000   10.0000    2.0000    1.0000
   10.0000   20.0000    2.0000    1.0000
   20.0000   20.0000    2.0000    1.0000
   10.0000   10.0000    4.0000    1.0000
   20.0000   10.0000    4.0000    1.0000
   10.0000   20.0000    4.0000    1.0000
   20.0000   20.0000    4.0000    1.0000
   15.0000   15.0000    5.0000    1.0000
   15.0000   15.0000    1.0000    1.0000
   15.0000   15.0000    3.0000    1.0000

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Ground truth
X =
    10    10    10     1
    20    10    10     1
    10    20    10     1
    20    20    10     1
    10    10    20     1
    20    10    20     1
    10    20    20     1
    20    20    20     1
    15    15    25     1
    15    15     5     1
    15    15    15     1
```

References

[1] An Efficient Solution to the Five-Point Relative Pose Problem, David Nister

[2] Multiview geometry in Computer vision, R. Hartley, A. Zisserman