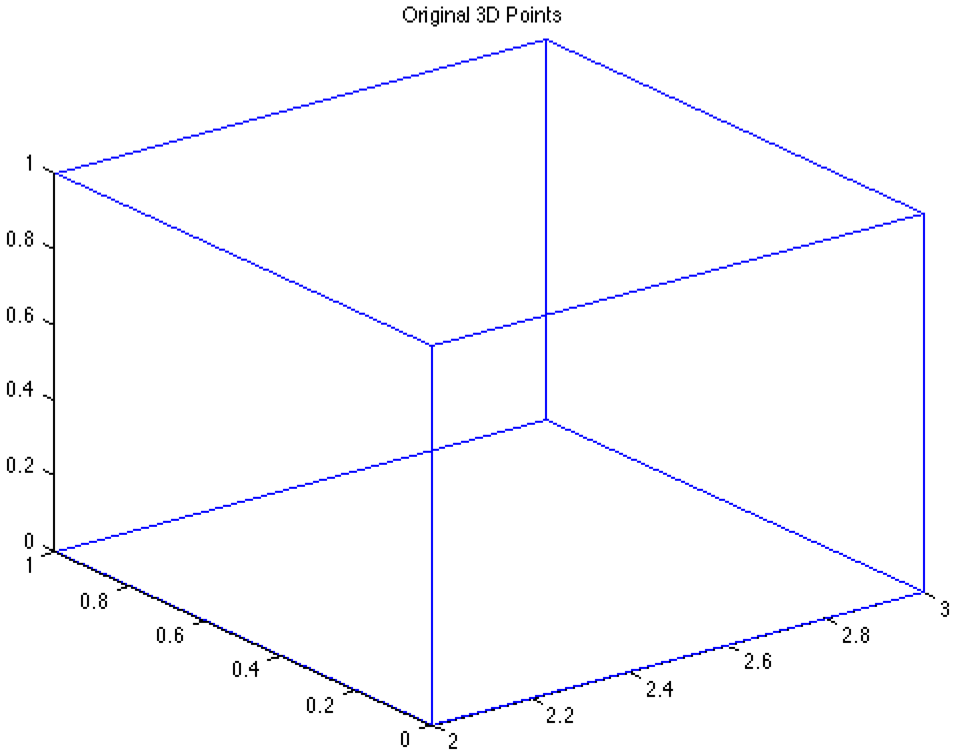
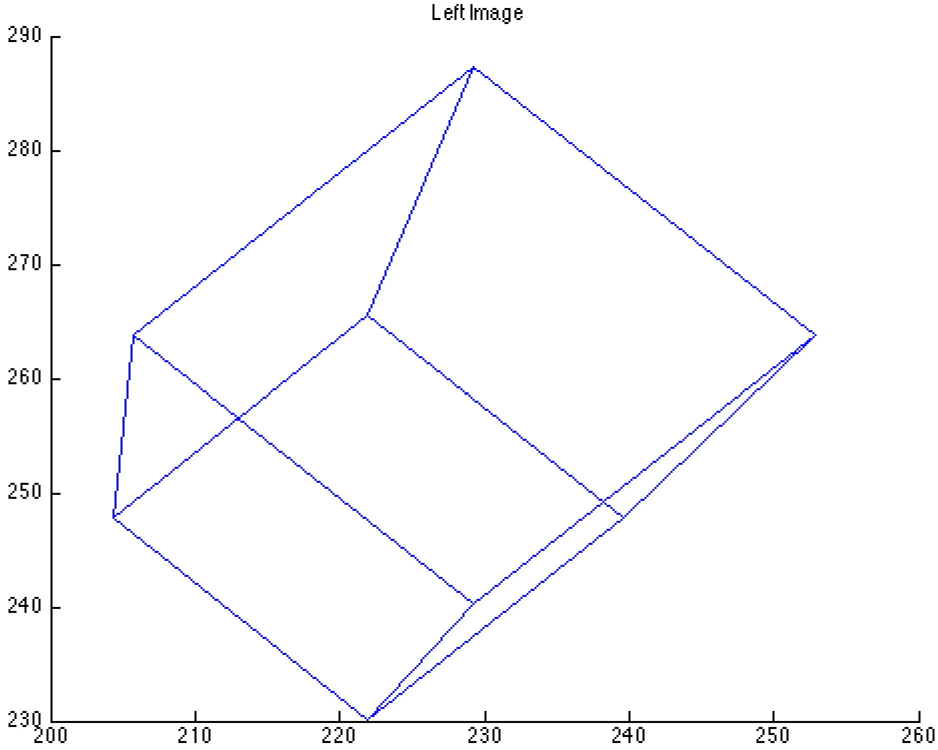
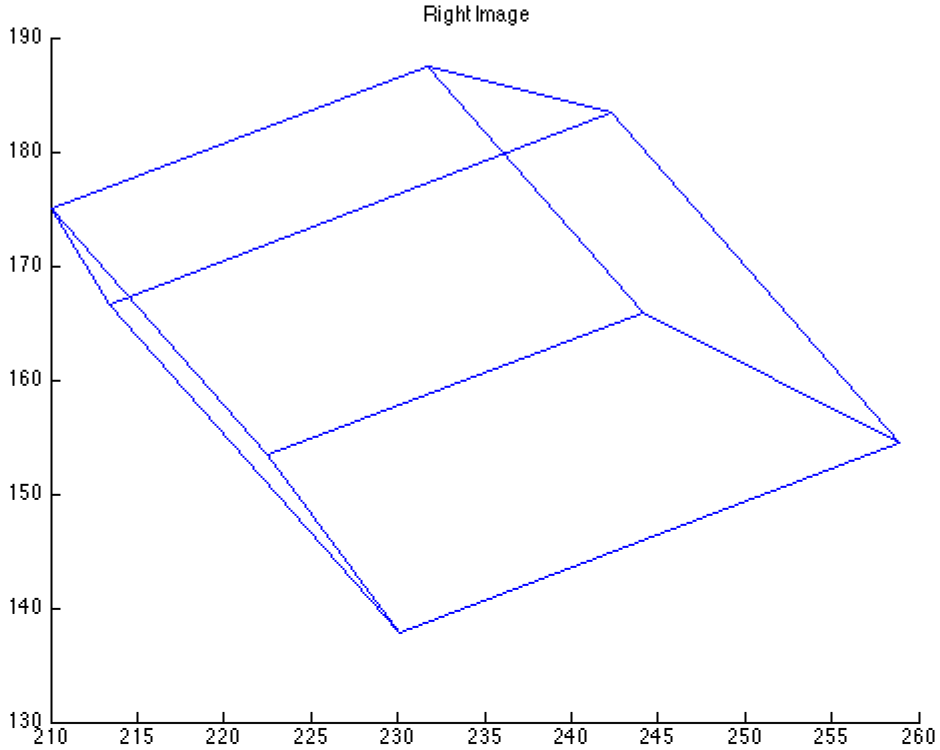
```
% intrinsic param
K = [-100 0 200 ;
    0 -100 200 ;
    0 0 1];

Mextleft = [ 0.707 0.707 0 -3 ;-0.707 0.707 0 -0.5; 0 0  1 3];
Mextright = [ 0.866 -0.5 0 -3 ;0.5 0.866 0 -0.5; 0 0 1 3];

pts = [ 2 0 0 ;
  3 0 0;
  3 1 0;
  2 1 0;
  2 0 1 ;
  3 0 1;
  3 1 1;
  2 1 1;
  2.5 0.5 2];
%for i = 1:9,
 %   pts(i,:) = rand(1,3);
%end;

%pts = [ 0 0 0 ; 0 1 0; 1 1 0; -1 1 0;1 0 1 ;1 0 1;1 1 1;1 1 1;1.5 0.5 2];

NN = 9;
pix = zeros(NN,3);
for i = 1:NN,
    pixels = K*Mextleft * [pts(i,1) pts(i,2) pts(i,3) 1]';
    leftpix(i,:) = pixels./pixels(3);
    pixels = K*Mextright * [pts(i,1) pts(i,2) pts(i,3) 1]';
    rightpix(i,:) = pixels./pixels(3);
end

% rightpix and leftpix are the list of corresponding points (attainable by
% ginput also

figure(1);clf;view(3)
drawmy3dobject(pts(:,1:3));title('Original 3D Points');

figure(2);clf;
drawmyobject(leftpix); title('Left Image');
figure(3);clf;
drawmyobject(rightpix); title('Right Image');
```


Original 3D Points
1
0.8
0.6
0.4
0.2
0
1
0.8
0.6
0.4
0.2
0
2
2.2
2.4
2.6
2.8
3


Left Image
290
280
270
260
250
240
230
200
210
220
230
240
250
260



From pixels to rays

```
rightray = inv(K)*[rightpix(:,1) rightpix(:,2) rightpix(:,3)]';
leftray = inv(K)*[leftpix(:,1) leftpix(:,2) leftpix(:,3)]';
```

STEREO RECTIFICATION With known camera matrices

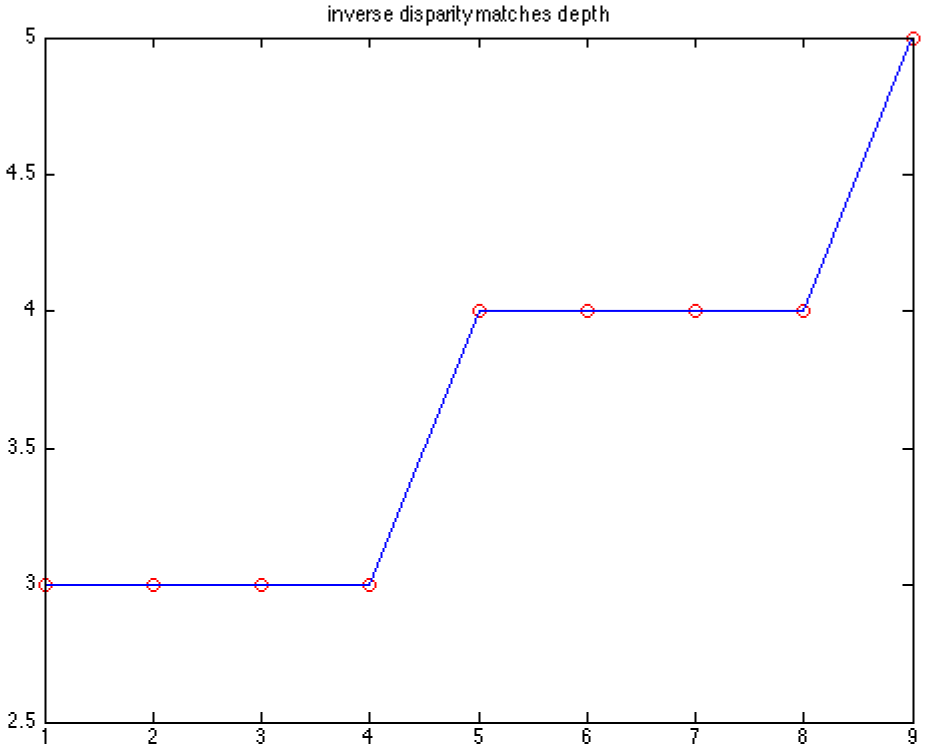
```
Trw = [Mextright ; 0 0 0 1];
Tlw = [Mextleft; 0 0 0 1];
Twr = inv(Trw); % can be done using transpose
Twl = inv(Tlw); % can be done using transpose

Tlr = Tlw*Twr;
% Rotation from right to left coordinate frame
Rlr = Tlr(1:3,1:3);
    % translation
tlr = Tlr(1:3,4);

% For rectification, we want the x axis of the new coordinage frame to be the vect
% optical axes, so we can write the first column as:
v1 = tlr./norm(tlr);
% The y axis should be perpendicular to the optical axis [0 0 1] and the x
% axis, so take the cross product
```

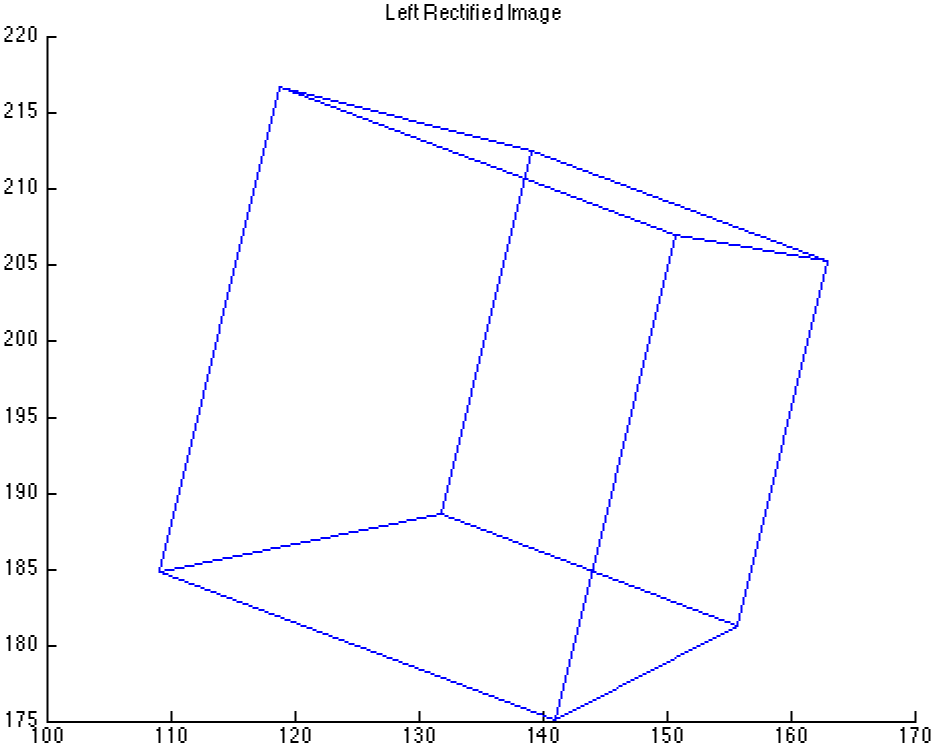
```
v2 = [-v1(2) v1(1) 0 ]';
v2 = v2./norm(v2);
% the new optical axis or z axis, must be perpendicular to the x and y axis
v3 = cross(v1,v2);
Rrectleft = [v1'; v2';v3';]'; % Now we can write the rotation matrix by interpreti
% to rectify the points in the right camera, first multiply by Rrect
% so that the relative orientation of the two coordinates is the same;
% then Rlr will align the right coordinate points with the left coordinate
% frame to make parallel cameras.

% Now the column vectors for the Rotation from the desired rectified frame to the
% frame. Transpose (inverse) to get the rotation from the left frame to the
% desired rectified frame
Rrectleft = Rrectleft';
Rrectright = Rlr*Rrectleft;


% Rectify by rotating the rays
ll = Rrectleft*leftray;
rr = Rrectright*rightray;
diff = ll-rr;
disparity = sqrt(diff(1,:).*diff(1,:) + diff(2,:).*diff(2,:))

% Compute Actual Depth ... for comparison
pts_wrt_camera = Tlw*[pts';ones(9,1)']; % transform from world to left camera coor
% depth is with respect to the left camera
% After this transformation the z componenent is the actual (groundtruth) depth
groundtruth = pts_wrt_camera(3,:);


% find the scale factor by using only one point
scale_factor = groundtruth(1)/(1.0./disparity(1));

% depth is inversely proportional to disparity. That means the
% disparity as computed with the rectified points should only be
% a scale factor from the actual depth. (Same scale factor for all points)

% Echo the two to visually compare
(1.0./disparity)*scale_factor
groundtruth
% comparison plot
figure(4);
plot((1.0./disparity)*scale_factor,'ro-');
hold on;
plot(groundtruth,'b-');
title('inverse disparity matches depth');

% For illustration, draw the rectified "images"
llpix = K*ll;
rrpix = K*rr;
figure(5);clf;
drawmyobject(llpix'); title('Left Rectified Image');
figure(6);clf;
drawmyobject(rrpix'); title('Right Rectified Image');
```

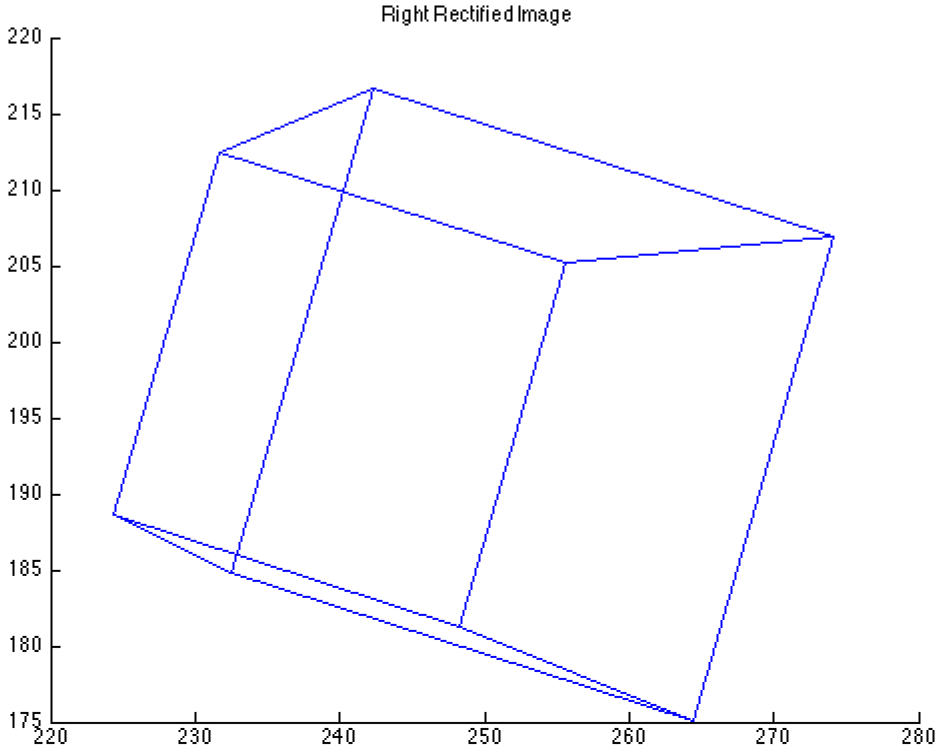
```
% Now draw them together on the same image so that one can observe
% that the y values of corresponding points are equal.
figure(7);clf;
drawmyobject(llpix'); hold on;
drawmyobject(rrpix'); title('Right and Left Rectified Image');
```

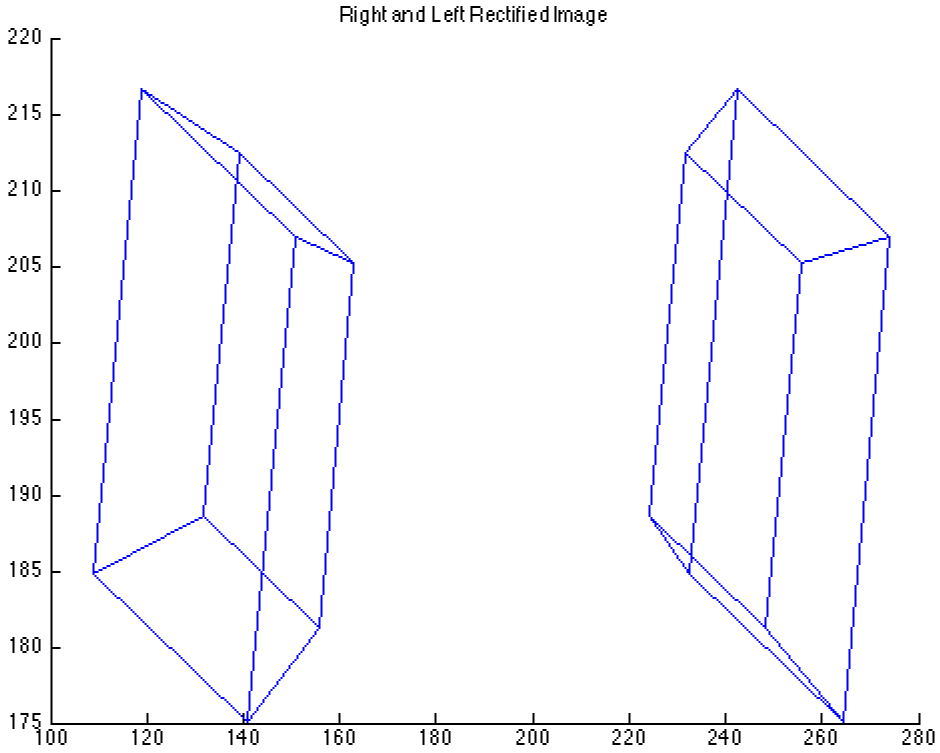
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now assume that no parameters are known and only point correspondences
% are given
% Reconstruct F from point correspondences.
%
% Using the epipolar constraint between the point correspondences, F can be
% found


for i = 1:NN
tt=leftpix(i,:)' * rightpix(i,:); % 3 x3 matrix
%form the matrix for Aq = 0, where q is 9x1 the elements of
A(i,:) = [tt(1,:) tt(2,:) tt(3,:)];
end;
```

```
disparity =

  Columns 1 through 7

    1.2342    1.2342    1.2342    1.2342    0.9257    0.9257    0.9257

  Columns 8 through 9

    0.9257    0.7405


ans =

  Columns 1 through 7

    3.0000    3.0000    3.0000    3.0000    4.0000    4.0000    4.0000

  Columns 8 through 9

    4.0000    5.0000


groundtruth =

     3     3     3     3     4     4     4     4     5
```


inverse disparity matches depth
5
4.5
4
3.5
3
2.5
1
2
3
4
5
6
7
8
9


Left Rectified Image
220
215
210
205
200
195
190
185
180
175
100
110
120
130
140
150
160
170


Right Rectified Image
220
215
210
205
200
195
190
185
180
175
220
230
240
250
260
270
280


Right and Left Rectified Image
220
215
210
205
200
195
190
185
180
175
100
120
140
160
180
200
220
240
260
280

Published with MATLAB® R2013b