Q1

```
im = imread('shoe.jpg');
imshow(im);
disp("Click four corner of five dollar bill");
[x,y] = getpts();

%  get actual size of five dolloar bill in mm
%

real_five_dollar_x = [1;70.85;70.85;1];
real_five_dollar_y = [1;1;153.4;153.4];


%
%  Compute Homography transformation
%


A = [];
for i = 1:size(x,1)
    A = [A;x(i,1) y(i,1) 1 0 0 0 -real_five_dollar_x(i,1)*x(i,1)
 -real_five_dollar_x(i,1)*y(i,1) -real_five_dollar_x(i,1);0
 0 0 x(i,1) y(i,1) 1 -real_five_dollar_y(i,1)*x(i,1) -
real_five_dollar_y(i,1)*y(i,1) -real_five_dollar_y(i,1)];

end

[U,S,V] = svd(A);
h = V(:,size(V,2));


%
%  click the shoe size
%

disp("click the length and width of shoe");
[shoe_x,shoe_y] = getpts();
shoe_new_position = zeros(3,2);
for idx = 1:size(shoe_y,1)
    new_x = floor((h(1,1)*shoe_x(idx,1) + h(2,1)*shoe_y(idx,1) +
 h(3,1))/(h(7,1)*shoe_x(idx,1)+h(8,1)*shoe_y(idx,1)+h(9,1)));
    new_y = floor((h(4,1)*shoe_x(idx,1)+h(5,1)*shoe_y(idx,1)+h(6,1))/
(h(7,1)*shoe_x(idx,1)+h(8,1)*shoe_y(idx,1)+h(9,1)));
    shoe_new_position(idx,1) = new_x;
    shoe_new_position(idx,2) = new_y;
end


%
%  display the result
%
```

```
d_length =
 pdist([shoe_new_position(1,:);shoe_new_position(2,:)],'euclidean');
X= ['Shoe Length: ', num2str(d_length)];
disp(X);
d_width =
 pdist([shoe_new_position(3,:);shoe_new_position(2,:)],'euclidean');
Y= ['Shoe Width: ', num2str(d_width)];
disp(Y);
```

```
Click four corner of five dollar bill
click the length and width of shoe
Shoe Length: 269.6702
Shoe Width: 71.8471
```

Published with MATLAB® R2017b

Q2a

```
%
%  Compute scale and rotation invariant keypoints in both images
%

    I = imread('book.jpg');
    Ib_orignal = imread('findbook.jpg');
    I_orignal = imread('book.jpg');
    Ib = imread('findbook.jpg');
    I = single(rgb2gray(I));
    Ib = single(rgb2gray(Ib));
    [f,d] = vl_sift(I);
    [fb, db] = vl_sift(Ib);
    d = double(d);
    db = double(db);
    euc= pdist2(d', db', 'euclidean');
    succssful_rate = 0.95;
    threashold = 0.6;


    I_points = [];
    Ib_points = [];


    %
    %  If ratio between closest and second closest match is < 0.6,
 keep match
    %

    for i = 1:size(euc,1)
        [first_min_value, first_min_idx] = min(euc(i,:));
        second_min_vaule = min(setdiff(euc(i,:),first_min_value));
        if first_min_value/second_min_vaule < threashold
            Ib_points = [Ib_points;fb(1,first_min_idx)
 fb(2,first_min_idx)];
            I_points = [I_points;f(1,i) f(2,i)];
        end
    end



    %
    %  Do RANSAC to compute affine transformation
    %

    com_list = combnk(1:size(I_points,1),3);
    C= zeros(size(com_list,1),size(com_list,2));
    rand = randperm(size(com_list,1));
    for idx = 1:size(com_list,1)
        C(idx,:) = com_list(rand(1,idx),:);
    end
```

```
   biggest_inliner = 0;
   Final_A = [];
   SSD = 0;

   %
   %  Compute the number of inliers
   %

   for idx = 1:size(C,1)
       temp_ssd = 0;
       P = [];
       P_Ib = [];
       for i=1:3
           P = [P; I_points(C(idx,i),1) I_points(C(idx,i),2) 0 0 1 0;
0 0 I_points(C(idx,i),1) I_points(C(idx,i),2) 0 1];
           P_Ib = [P_Ib;Ib_points(C(idx,i),1);
Ib_points(C(idx,i),2)];
       end
       A = pinv(P)*P_Ib;
       in_liner = 0;
       for point = 1:size(I_points,1)
           test_P = [I_points(point,1) I_points(point,2) 0 0 1 0; 0 0
I_points(point,1) I_points(point,2) 0 1];
           test_result = test_P * A;
           temp_ssd = temp_ssd +
pdist2(test_result',Ib_points(point,:), 'euclidean')^2;
           if(pdist2(test_result',Ib_points(point,:), 'euclidean')<3)
               in_liner = in_liner + 1;
           end
       end

       %
       %  Find A that gave the most inliers
       %

       if(in_liner>biggest_inliner)
           Final_A = A;
           biggest_inliner = in_liner;
           SSD = temp_ssd;
       end
       if(biggest_inliner > size(I_points,1)*0.95)
           break;
       end
   end

   book_shap = [0 0 0 0 1 0; 0 0 0 0 0 1; 0 size(I,1) 0 0 1 0; 0 0
0 size(I,1) 0 1; size(I,2) 0 0 0 1 0;0 0 size(I,2) 0 0 1;size(I,2)
size(I,1) 0 0 1 0;0 0 size(I,2) size(I,1) 0 1];

   new_position = book_shap * A;

   fg = figure;imagesc(Ib_orignal);axis image;hold on;colormap gray;
   for idx = 1:2:size(new_position,1)
```

```
    plot(new_position(idx,1),new_position(idx+1,1),'ob');
end
hold off;
```

Published with MATLAB® R2017b

Q2b

```
Ib = imresize(imread('mugShot.jpg'),0.25);
cuts = dir('./shredded/*.png');

% reassembling the image in random permutations
order = perms(1:size(cuts));
SSD_LIST = [];

%Rank your models by the mean residual SSD.
for i = 1:size(order,1)
    I = imresize([imread(strcat('./shredded/',cuts(order(i,1)).name))
 imread(strcat('./shredded/',cuts(order(i,2)).name))
 imread(strcat('./shredded/',cuts(order(i,3)).name))
 imread(strcat('./shredded/',cuts(order(i,4)).name))
 imread(strcat('./shredded/',cuts(order(i,5)).name)) imread(strcat('./
shredded/',cuts(order(i,6)).name))],0.25);

    [FINAL_A,biggest_inliner,SSD] = affine(I,Ib);
    SSD_LIST = [SSD_LIST SSD];
end

[value, index] = max(SSD_LIST(1,:));
Final_I = [imread(strcat('./shredded/',cuts(order(index,1)).name))
 imread(strcat('./shredded/',cuts(order(index,2)).name))
 imread(strcat('./shredded/',cuts(order(index,3)).name))
 imread(strcat('./shredded/',cuts(order(index,4)).name))
 imread(strcat('./shredded/',cuts(order(index,5)).name))
 imread(strcat('./shredded/',cuts(order(index,6)).name))];
imshow(Final_I);
```

Q3

October 28, 2017 6:41 PM

a.

No. you can't estimate the distance between railway in world coordinates.

Reason:

from world coordinates to pixel coordinates, we have:

$$\begin{bmatrix} ax \\ ay \\ a \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

So, if we want to get the world coordinates from pixel coordinates, we have to do backward projection.

because $P = \begin{bmatrix} f & 0 & P_x \\ 0 & f & P_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} I_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}$

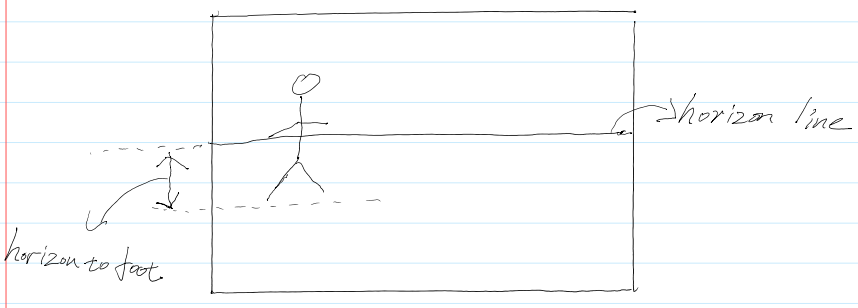
☆ P is not Invertible

Therefore, we can't get world coordinates.

b.

Yes. From Lecture 10, we know that "Same distance as where the horizon interseets a building"

And we knew that Camera centre is 95 cm above ground.



So we have the following equation:

$$\frac{man\_totall\_pixel\_height}{horizon\_to\_foot\_pixel\_height} = \frac{man\_actual\_height}{95}$$

$$\therefore man\_actual\_height = \frac{95(man\_totall\_pixel\_height)}{horizon\_to\_foot\_pixel\_height.}$$

Q4

October 28, 2017 4:22 PM

a. $\because K = \begin{bmatrix} f & 0 & P_x \\ 0 & f & P_y \\ 0 & 0 & 1 \end{bmatrix}$

$\because f = 721.5$

$(P_x, P_y) = (609.6, 172.9)$

$\therefore K = \begin{bmatrix} 721.5 & 0 & 609.6 \\ 0 & 721.5 & 172.9 \\ 0 & 0 & 1 \end{bmatrix}$

b.

because Camera is 1.7 meters above ground. and 1.7 meter = 1700 mm

So,

Equation of the ground:

$$0X + 0Y + 0Z + 1700 = 0$$

C.

$\because K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} w \cdot x \\ w \cdot y \\ w \end{bmatrix}$

$\therefore \begin{bmatrix} f & 0 & P_x \\ 0 & f & P_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix}$

$\therefore fX + ZP_x = wx$

$f \cdot Y + ZP_y = wy$

$Z = w$

$\therefore fY + ZP_y = Zy$

$fY = Z(y - P_y)$

$\rightarrow - \quad \perp Y$

$$Z = \frac{fY}{y - P_y}$$

$$fX + ZP_x = wx$$

$$fX + ZP_x = Z_x$$

$$fX + \frac{fYP_x}{y - P_y} = \frac{fY_x}{y - P_y}$$

$$fX = \frac{fY(x - P_x)}{y - P_y}$$

$$X = \frac{Y(x - P_x)}{y - P_y}$$

$$\therefore \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \frac{-1700x + 1036320}{y - 609.6} \\ -1700 \\ \frac{-1226550}{y - 609.6} \end{bmatrix}$$

Bonus 1

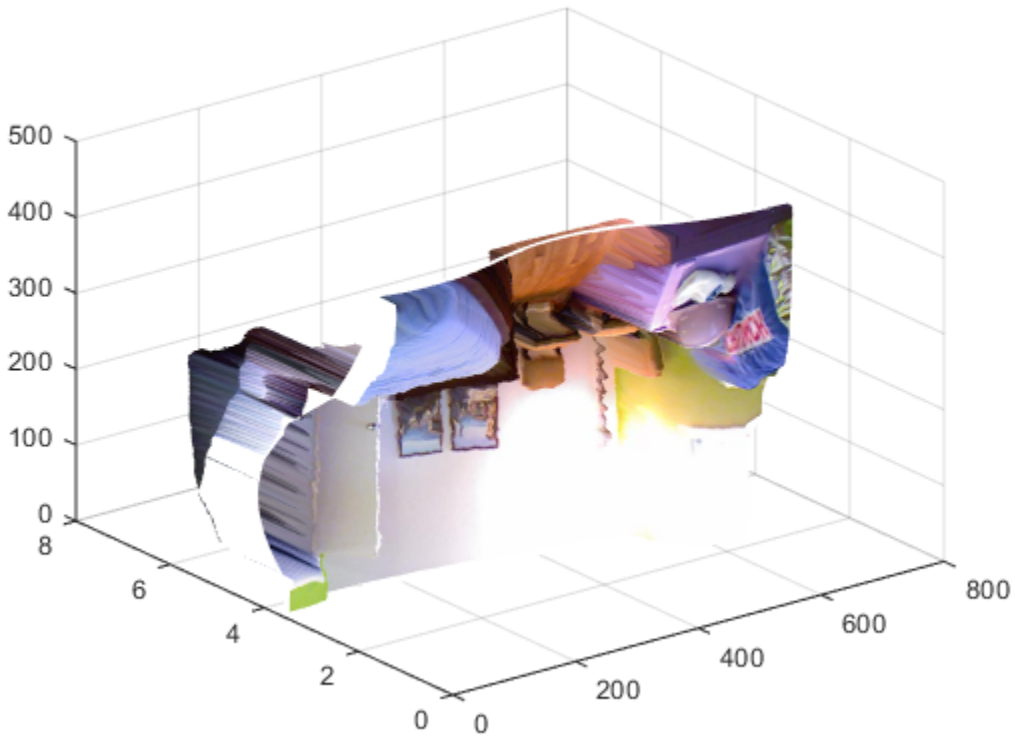
```
S = load('rgbd.mat');
length = [1:size(S.im,2)];
color = rgb2gray(S.im);
width = zeros(480,640);

for i = 2:size(S.im,1)
    length = [length; [1:size(S.im,2)]];
end

for j = 1:size(S.im,2)
        width(:,j) = [1:size(S.im,1)];
end

color = S.im;

figure,surf(length,S.depth,width,S.im,'EdgeColor','None');
```



Published with MATLAB® R2017b

500
450
400
350
300
250
200
150
100
50
0
8
6
4
2
0
0
100
200
300
400
500
600
700