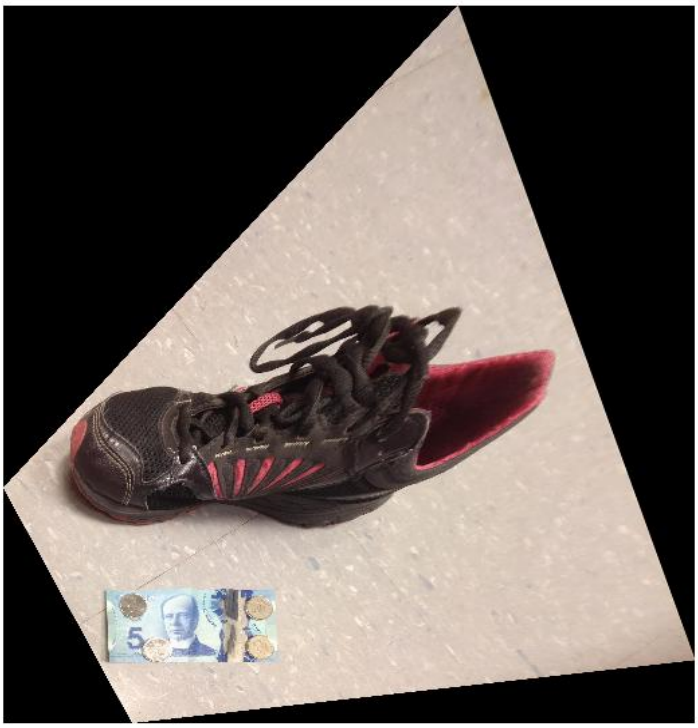
1.
Length = 29cm~
Width = 11cm~



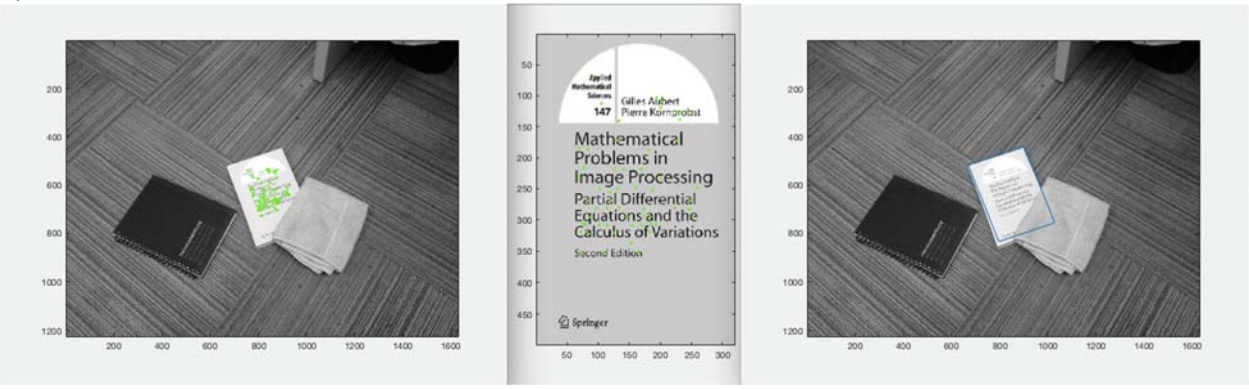
```
%set the corners of the $5 bill
 x = [775.5 1067.5 1367.5 1159.5]';
 y = [985.5 509.5 587.5 1147.5]';

%actual size of $5 in pixels as mm
x2 = [1, 1524, 1524, 1]';
y2 = [1, 1, 699, 699]';

% compute homography
tform = maketform('projective',[x,y],[x2,y2]);

% warp the image according to homography
[imrec] = imtransform(im, tform, 'bicubic', 'XYScale',1);

mshow(imrec)
disp('pick two points for length, double click on second point');
[lx,ly] = getpts();
disp('pick two points for width, double click on second point');
[wx,wy] = getpts();

length = sqrt((lx(1) - lx(2))^2 + (ly(1) - ly(2))^2)/100;
width = sqrt((wx(1) - wx(2))^2 + (wy(1) - wy(2))^2)/100;
```

2.
a)

b) If the camera is 95cm off the ground and the ground is planar, we know the horizon is 95cm off the ground at any point in the image. So we can compute the pixel distance from the ground to the horizon near the man and then use the ratio the estimate the height of the man.
Pixel height near man from ground to horizon = 165pixels
Ratio of pixels to cm = 95/165
Height of man in pixels = 285
Height of man = 285*(95/165) = 162.5

3.

a)




```
function matching_points = get_matches(I, Ib, threshold)

I = single(I) ;
Ib = single(Ib) ;

[f,d] = vl_sift(I) ;
[fb, db] = vl_sift(Ib) ;

d = double(d);
db = double(db);
euc= pdist2(d', db', 'euclidean');
sorted = sort(euc, 2);
ratios=sorted(:,1)./sorted(:,2);

matches = zeros(size(find(ratios<=threshold),1), 3);
for i = 1:size(euc,1)
    if ratios(i) < threshold
        matches(i,1) = ratios(i);
        matches(i,2)= i;
        matches(i,3)=find(euc(i,:)==sorted(i,1));
    end
end

matches( ~any(matches,2), : ) = [];
matching_points = zeros(size(matches,1), 5);

for i = 1:size(matches,1)
    matching_points(i,1) = matches(i,1);
    matching_points(i,2:3) = [f(1,matches(i,2)) f(2,matches(i,2))];
    matching_points(i,4:5) = [fb(1,matches(i,3)) fb(2,matches(i,3))];

end

end
```

```
function [good_points, good_affine] = affine_t(I, Ib, threshold)
matching_points = get_matches(I,Ib , threshold);

mmk = matching_points';
figure, imagesc(I), axis image, colormap(gray),hold on
plot(mmk(2,:),mmk(3,:),'g.') ;
hold off;
figure, imagesc(Ib), axis image, colormap(gray),hold on
plot(mmk(4,:),mmk(5,:),'g.') ;
hold off;

% format of matching points
% ratio | xcoord-img1 | ycoord-img1 | xcoord-img2 | ycoord-img2

matching_pointss = sortrows(matching_points, 1);
k = size(matching_pointss, 1);

inl_thr=5;
max_matches = 0;
S = log(1-0.9)/log(1-(1/(size(matching_pointss, 1)/3))^3)
indices = [];
% RANSAC
while (S>0 && k >=3)
    random_3 = randperm(k,3);
    rand_points = matching_pointss(random_3,:);
    S = S-1;

    P = [];
    Pp = [];

    for i = 1:3
        x1 = rand_points(i,2);
        y1 = rand_points(i,3);
        x2 = rand_points(i,4);
        y2 = rand_points(i,5);

        P(size(P,1)+1,:) = [x1 y1 0 0 1 0];
        P(size(P,1)+1,:) = [0 0 x1 y1 0 1];

        Pp(size(Pp,1)+1,:) = x2;
        Pp(size(Pp,1)+1,:) = y2;
    end

    %compute affine transformation for 3 random points
    penny = P'*inv(P*P');
    affine = penny*Pp;

    O = [];
    Op = [];
    for i = 1:size(matching_pointss, 1);
        x1 = matching_pointss(i,2);
        y1 = matching_pointss(i,3);
        x2 = matching_pointss(i,4);
```

```
        y2 = matching_pointss(i,5);

        O(size(O,1)+1,:) = [x1 y1 0 0 1 0];
        O(size(O,1)+1,:) = [0 0 x1 y1 0 1];

        Op(size(Op,1)+1,:) = x2;
        Op(size(Op,1)+1,:) = y2;
    end

    transform = O*affine;
    tformxy = [transform(1:2:length(transform))
transform(2:2:length(transform))];
    Opxy = [Op(1:2:length(Op)) Op(2:2:length(Op))];
    diff = abs(Opxy - tformxy);
    diff = diff(:,1) + diff(:,2);

    close = find(diff<=inl_thr);

    if max_matches < size(close, 1);
        max_matches = size(close,1);
        good_affine = affine;
        indices = close;
    end


end

if  max_matches <=3
    good_points = [];
    good_affine = [];
elseif ~isempty(indices)
    good_points = matching_pointss(indices,:);
end

end
```

```
I = rgb2gray(imread('book.jpg'));
Ib = rgb2gray(imread('findBook.jpg'));

[good_points, good_affine] = affine_t(I, Ib, 0.6);

mmk = good_points';
figure, imagesc(I), axis image, colormap(gray),hold on
plot(mmk(2,:),mmk(3,:),'g.') ;
hold off;
figure, imagesc(Ib), axis image, colormap(gray),hold on
plot(mmk(4,:),mmk(5,:),'g.') ;
hold off;

O = [];
Op = [];
corners = [1 1; size(I,2) 1; size(I,2) size(I,1);1 size(I,1)];
for i = 1:size(corners, 1);
    x1 = corners(i,1);
    y1 = corners(i,2);

    O(size(O,1)+1,:) = [x1 y1 0 0 1 0];
    O(size(O,1)+1,:) = [0 0 x1 y1 0 1];

end

transform = O*good_affine;

yr = transform(2:2:length(transform));
xr = transform(1:2:length(transform));
xr= [xr' xr(1)];
yr= [yr' yr(1)];

figure, imagesc(Ib), axis image, colormap(gray),hold on
plot(xr,yr);
hold off;
```

b)

```
filenames = dir('./shredded/*.png');
reconstruction = [];
nulls = 0;
max_points = 0;
two_combo = repmat([0 {'1'} {'1'}], 6, 1);
Ib = imread('mugShot.jpg');
Ib = rgb2gray(Ib);

%computes best pairs of shredded peices by comparing all possible
%permutations of 2 images with mugshot and ranking them by number of
%matches
for i = 1:size(filenames,1)
    max_points = 0;
    for j = 1:size(filenames,1)
        if ~strcmp(filenames(i).name, filenames(j).name)
            I = [imread(strcat('./shredded/',filenames(i).name))
imread(strcat('./shredded/',filenames(j).name))];
            I = rgb2gray(I);

            [good_points, good_affine] = affine_t(I, Ib, 0.75)


            if max_points < size(good_points, 1)
                max_points = size(good_points, 1)
                two_combo(i,:) = [max_points
{strcat('./shredded/',filenames(i).name)}
{strcat('./shredded/',filenames(j).name)}];
            end
        end
    end

end

%takes the two image matche with the highest rank and attemps to append
%other matches onto it using the rank and shred name
two_combo = sortrows(two_combo, 1);
two_combo = flipud(two_combo);
best = two_combo(1,2:3);
points =0;
for i = 2:size(two_combo,1)
    %left side
    if ~strcmp(two_combo{i,2}, best{size(best,2)})
        if strcmp(two_combo{i,3}, best{1})
            if points < two_combo{i,1}
                points = two_combo{i,1};
                left = two_combo(i,2);
            end
        end
    end
end
for i =1:size(best,2)
    if strcmp(best{i}, left)
        left = [];
```

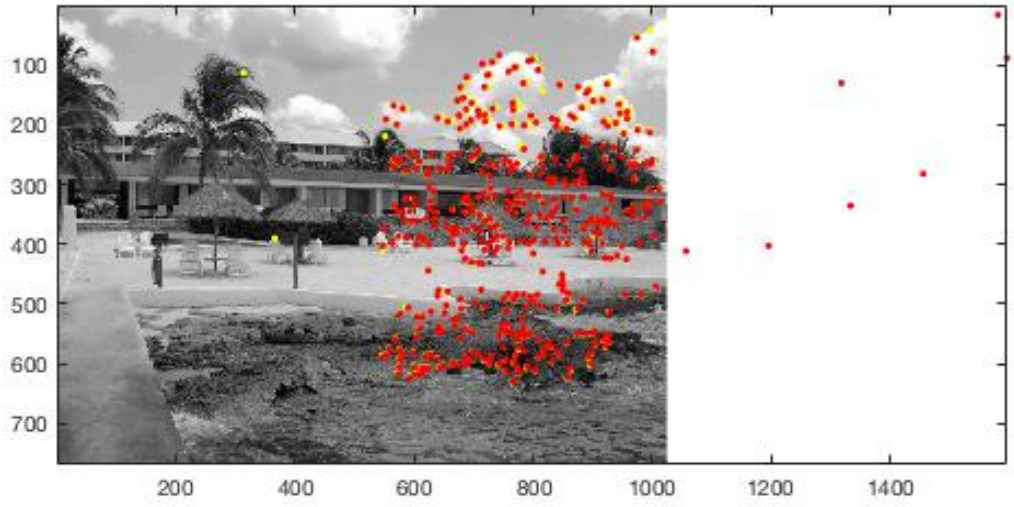
```
    end
end
best = [left best];

points =0;
for i = 2:size(two_combo,1)
    %right side
    if ~strcmp(two_combo{i,3}, best{1})
        if strcmp(two_combo{i,2}, best{size(best,2)})
            if points < two_combo{i,1}
                points = two_combo{i,1};
                right = two_combo(i,3);
            end
        end
    end
end
for i =1:size(best, 2)
    if strcmp(best{i}, right)
        right = [];
    end
end
best = [best right];
I = [];
for i =1:size(best, 2)
    I = [I imread(best{i})];
end
imagesc(I)
```

The way I implemented the reconstruction algorithm, it uses ransac to find the best matching shreds compared to mugshot. So my algorithm only knows how to look for pieces that are part of mugshot. The other 3 shreds didn't have any part of mugshot in them, therefore the algorithm is unable to stich them to the image at all. If I used random permutations those 3 pieces would be in a random order anyways. (I tried random, but stopped in fear that my laptop might spontaneously combust).

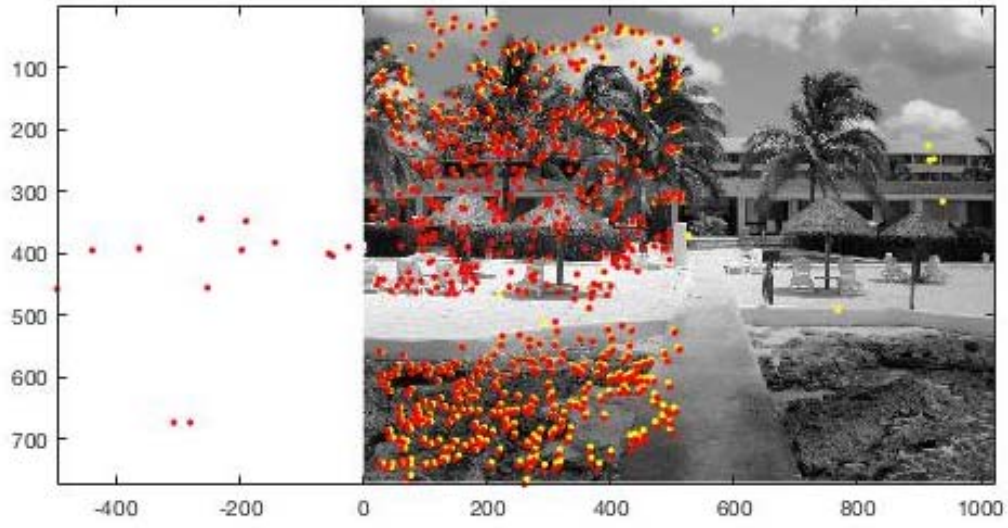
4.

a)

Matching keypoints between Hotel-02 and Hotel-01, 02 points in Yellow and points transformed from 01 in Red.



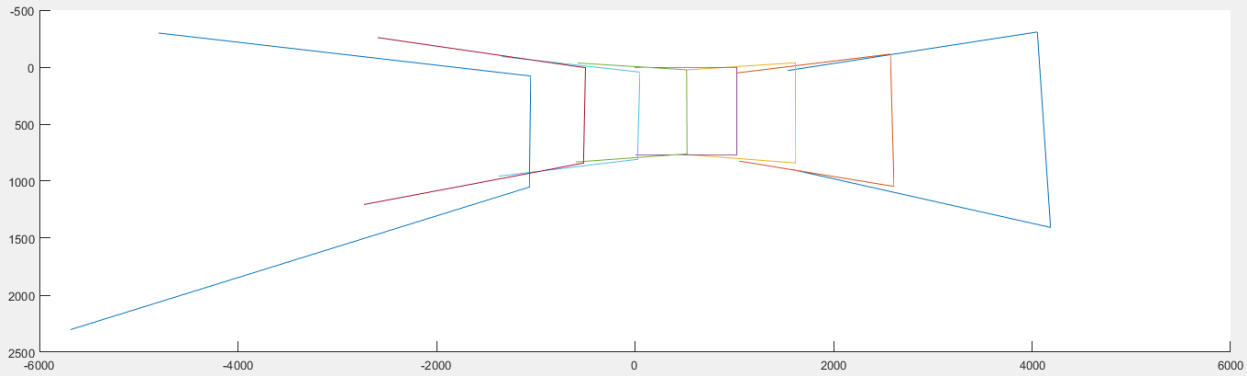
Same colour scheme, between Hotel-03 and 04.




```
next = imread(strcat('./hotel/',char(names(i))));
current = imread(strcat('./hotel/',char(names(i+1))));

matching_points = get_matches(next, current, 0.7);
matching_pointss = sortrows(matching_points, 1);
points = matching_pointss(1:15,:);

%use top k correspondeces to compute homography
H = compute_homography([points(:,2) points(:,3)], [points(:,4) points(:,5)]);
H_s = [H(1:3)';H(4:6)';H(7:9)'];
```

```
%4a
fourA(H_s, matching_pointss, current);

function fourA(H_s, matching_pointss, current)

    out = zeros(size(matching_pointss,1),3);
    for i = 1:size(matching_pointss,1)
        out(i,:) = (H_s*[matching_pointss(i,2);matching_pointss(i,3);1])';
        out(i,:) = out(i,:)./out(i,3);
    end

    figure, imagesc(current), axis image, colormap(gray),hold on
        plot(matching_pointss(:,4),matching_pointss(:,5), 'y.');
        plot(out(:,1),out(:,2), 'r.');
        hold off;

end

function H = compute_homography(points1, points2)
    A =[];
    for i = 1:size(points1);
        x1 = points1(i,1);
        y1 = points1(i,2);
        x2 = points2(i,1);
        y2 = points2(i,2);

        A(size(A,1)+1,:) = [x1 y1 1 0 0 0 -x2*x1 -x2*y1 -x2];
        A(size(A,1)+1,:) = [0 0 0 x1 y1 1 -y2*x1 -y2*y1 -y2];

    end
    [U, S, V] = svd(A);
    H = V(:,9);
end
```

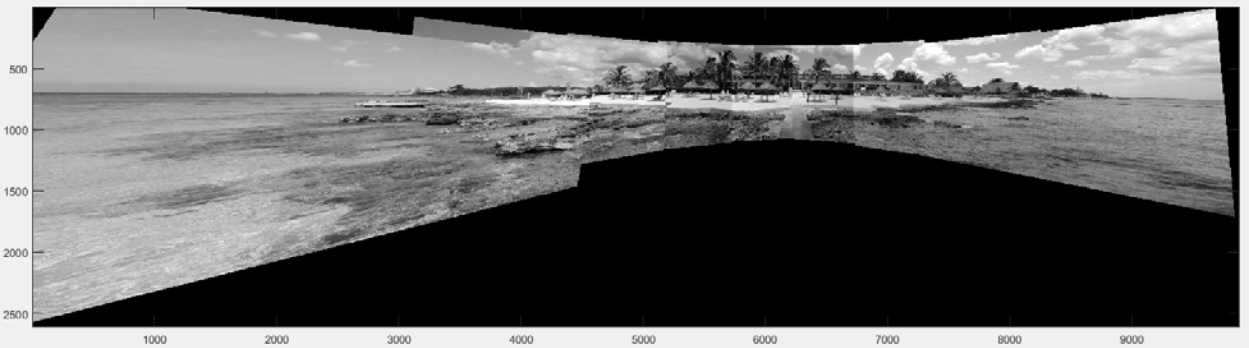
b)




```
filenames = dir('./hotel/*.png');
names = struct2cell(filenames);
names = names(1,:);
mid = size(names,2)/2;
middle = imread(strcat('./hotel/',char(names(mid))));
corners = repmat([1 1; 1024 1; 1024 768;1 768],size(names));
Homos = zeros(9, size(names,2));
for i = 1:mid-1

    next = imread(strcat('./hotel/',char(names(i))));
    current = imread(strcat('./hotel/',char(names(i+1))));

    matching_points = get_matches(next, current, 0.7);
    matching_pointss = sortrows(matching_points, 1);
    %k = size(matching_pointss, 1);
    points = matching_pointss(1:15,:);

    xI = points(:,2);
    yI = points(:,3);
    xB = points(:,4);
    yB = points(:,5);

    %use top k correspondeces to compute homography
    H = compute_homography([points(:,2) points(:,3)], [points(:,4)
points(:,5)]);
    Homos(:,i) = H;
    H_s = [H(1:3)';H(4:6)';H(7:9)'];

    %4a
    fourA(H_s, matching_pointss, current);

    out = zeros(size(corners,1),3);
    for x = 1:i
        for j = 1:size(corners,1)
            out(j,:) = (H_s*[corners(j,(x*2)-1);corners(j,x*2);1])';
            out(j,:) = out(j,:)./out(j,3);
        end
        corners(:,(x*2)-1:x*2) = out(:,1:2);
    end
```

```
end

for i = size(names,2):-1:mid+1

    next = imread(strcat('./hotel/',char(names(i))));
    current = imread(strcat('./hotel/',char(names(i-1))));

    matching_points = get_matches(next, current, 0.7);
    matching_pointss = sortrows(matching_points, 1);
    %k = size(matching_pointss, 1);
    points = matching_pointss(1:15,:);

    xI = points(:,2);
    yI = points(:,3);
    xB = points(:,4);
    yB = points(:,5);

    H = compute_homography([points(:,2) points(:,3)], [points(:,4)
points(:,5)]);

      %this is for use later to stitch the panorama
    Homos(:,i) = H;

    H_s = [H(1:3)';H(4:6)';H(7:9)'];

    %4a
    fourA(H_s, matching_pointss, current);

    out = zeros(size(corners,1),3);
    for x = size(names,2):-1:i
        for j = 1:size(corners,1)
            out(j,:) = (H_s*[corners(j,(x*2)-1);corners(j,x*2);1])';
            out(j,:) = out(j,:)./out(j,3);
        end
        corners(:,(x*2)-1:x*2) = out(:,1:2);
    end
end


figure, axis ij, hold on
for i = 1:size(corners,2)/2
    plot(corners(:,i*2-1), corners(:,i*2));
end
hold off;
```

c) `figure, imagesc(panorama), axis ij, colormap(gray)`




```
%create list of transformations from the middle out
tforms(size(names)) = projective2d(eye(3));
for i = mid+1:size(Homos,2)
    H = Homos(:,i);
    H_s = [H(1:3)';H(4:6)';H(7:9)'];
    tforms(i) = projective2d(H_s');
    tforms(i).T = tforms(i-1).T * tforms(i).T;
end
for i = mid-1:-1:1
    H = Homos(:,i);
    H_s = [H(1:3)';H(4:6)';H(7:9)'];
    tforms(i) = projective2d(H_s');
    tforms(i).T = tforms(i+1).T * tforms(i).T;
end

xvals = corners(:,1:2:15);
yvals = corners(:,2:2:16);

xLimits = [floor(min(xvals(:))) ceil(max(xvals(:)))];
yLimits = [floor(min(yvals(:))) ceil(max(yvals(:)))];

width  = round(xLimits(2) - xLimits(1));
height = round(yLimits(2) - yLimits(1));

panoramaView = imref2d([height width], xLimits, yLimits);
panorama = zeros([height width], 'like', middle);

blender = vision.AlphaBlender('Operation', 'Binary mask', ...
    'MaskSource', 'Input port');

for i = 1:size(names,2)
    I = imread(strcat('./hotel/',char(names(i))));
    warpedImage = imwarp(I, tforms(i), 'OutputView', panoramaView);

    % Generate a binary mask.
    mask = imwarp(true(size(I,1),size(I,2)), tforms(i), 'OutputView',
panoramaView);

    % Overlay the warpedImage onto the panorama.
    panorama = step(blender, panorama, warpedImage, mask);
end
```

The code to display the panorama was taken from:
https://www.mathworks.com/examples/matlab-computer-vision/mw/vision_product-FeatureBasedPanoramicImageStitchingExample-feature-based-panoramic-image-stitching
With some modifications.