# ECE 661 Homework 2

Zeeshan Nadir
email: **znadir@purdue.edu**

Due Date: September 11, 2014

## Finding the Homography between two planes

The transformation from the world plane to the image plane can be written as

$$\mathbf{x}_i = H\mathbf{x}_w \tag{1}$$

where all the vectors and matrices are mentioned in homogeneous coordinates. The subscript $i$ refers to image plane and subscript $w$ refers to world plane. Writing eq. (1) in explicit format we get

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} \tag{2}$$

where we have set $h_{33} = 1$ and $z_w = 1$ because only ratios matter in homogeneous coordinates. The physical coordinates in the image plane are given by $(x_i', y_i') = (\frac{x_i}{z_i}, \frac{y_i}{z_i})$. If we establish a correspondence between a point in the world plane, and a point in the image plane then we shall get two equations corresponding to the physical $x$-coordinate and the physical $y$-coordinate. Since the total number of unknowns in eq. (2) are 8, therefore we require a minimum of 4 point correspondences to find the required homography.

The two equations that we get for each point to point correspondence are given as

$$x_i' = \frac{x_i}{z_i} = \frac{h_{11}x_w + h_{12}y_w + h_{13}}{h_{31}x_w + h_{32}y_w + h_{33}} \tag{3}$$

$$y_i' = \frac{x_i}{z_i} = \frac{h_{21}x_w + h_{22}y_w + h_{23}}{h_{31}x_w + h_{32}y_w + h_{33}} \tag{4}$$

We can write the above set of equations in form of $Ax = b$. Here $A \in \mathbb{R}^{2p\times 8}$ is the coefficient matrix, $x \in \mathbb{R}^8$ is vector that has elements of $H$, and $b \in \mathbb{R}^8$ is a vector of constants and $p$ is the number of point correspondences we have used. If we use 4 point correspondences, we get following equation in vector-matrix form

$$\begin{bmatrix}
x_w^{(1)} & y_w^{(1)} & 1 & 0 & 0 & 0 & -x_i' x_w^{(1)} & -x_i' y_w^{(1)} \\
0 & 0 & 0 & x_w^{(1)} & y_w^{(1)} & 1 & -y_i' x_w^{(1)} & -y_i' y_w^{(1)} \\
x_w^{(2)} & y_w^{(2)} & 1 & 0 & 0 & 0 & -x_i' x_w^{(2)} & -x_i' y_w^{(2)} \\
0 & 0 & 0 & x_w^{(2)} & y_w^{(2)} & 1 & -y_i' x_w^{(2)} & -y_i' y_w^{(2)} \\
x_w^{(3)} & y_w^{(3)} & 1 & 0 & 0 & 0 & -x_i' x_w^{(3)} & -x_i' y_w^{(3)} \\
0 & 0 & 0 & x_w^{(3)} & y_w^{(3)} & 1 & -y_i' x_w^{(3)} & -y_i' y_w^{(3)} \\
x_w^{(4)} & y_w^{(4)} & 1 & 0 & 0 & 0 & -x_i' x_w^{(4)} & -x_i' y_w^{(4)} \\
0 & 0 & 0 & x_w^{(4)} & y_w^{(4)} & 1 & -y_i' x_w^{(4)} & -y_i' y_w^{(4)}
\end{bmatrix}
\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix}
=
\begin{bmatrix} x'^{(1)} \\ y'^{(1)} \\ x'^{(2)} \\ y'^{(2)} \\ x'^{(3)} \\ y'^{(3)} \\ x'^{(4)} \\ y'^{(4)} \end{bmatrix} \tag{5}$$

Solving eq. (5) gives us the matrix $H$. In case if we have more than 4 point correspondences then we can use least squares estimate of $x$ i.e. $x = (A^T A)^{-1} A^T b$.

# Transformation of Image Plane to World Plane

Once we have solved for $H$ matrix, we can find its inverse $H^{-1}$ which maps the image plane back to the world plane. Using the boundary of the image and $H^{-1}$ matrix, we can find an encompassing rectangle in the world plane, that encompasses the image in the world plane. Using this rectangular boundary, we can find the width and height of the image in the world plane. The number of pixels in the rows of image in the world plane can be slected the same as in the image plane it self. The number of pixels in the columns of the image in the world plane can be selected according to the aspect ratio of the image in the world plane.

Once we have formed a grid of points in the world plane, then for each point in the world plane, we can use matrix $H^{-1}$ to find the corresponding point in the image plane. However when we apply $H^{-1}$ to points of world plane to find the corresponding points in the image plane, the resultant points in the image plane may not come out to be integers. Therefore we use bilinear interpolation to tackle this.

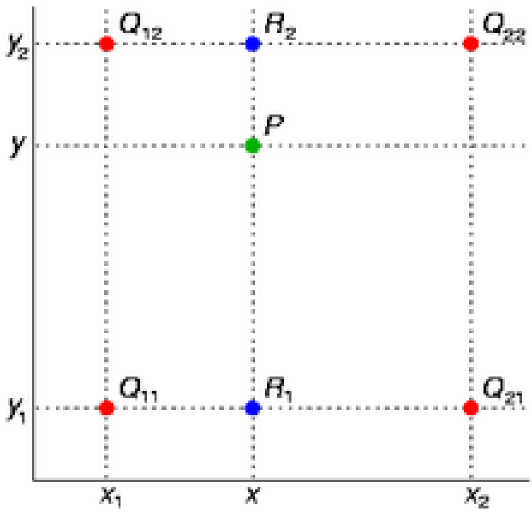


**Figure 1:** Bilinear Interpolation.

Bilinear interpolation is extension of linear interpolation for functions of two variables. The main idea here is to first interpolate in $x$ direction and then use those values to interpolate in $y$ direction. The scenario of bilinear interpolation is explained with the help of an example.

Suppose $P = (x, y)$ is a point in the image plane where $x$ and $y$ may not be integers. Suppose $(x_1, y_1)$, $(x_1, y_2)$, $(x_2, y_1)$ and $(x_2, y_2)$ are the nearest points that have integer as the coordinate values i.e. all these points are in $\mathbb{Z} \times \mathbb{Z}$ (shown in red in Fig. 1). Then we can find the value of function $f$ at $(x, y)$ as

$$\begin{aligned} f(x,y) = &\frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} f(x_1, y_1) + \frac{(x_1 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} f(x_2, y_1) \\ &+ \frac{(x_2 - x)(y_1 - y)}{(x_2 - x_1)(y_2 - y_1)} f(x_1, y_2) + \frac{(x_1 - x)(y_1 - y)}{(x_2 - x_1)(y_2 - y_1)} f(x_2, y_2) \end{aligned} \tag{6}$$

The above formula is obtained by first performing an interpolation at $R_1$ and $R_2$ (shown as blue points) in Fig. 1 and then using those two values to interpolate at $P$. Even though each of the basic step is linear (which are not shown here) in the sampled values and in the position, however the interpolation as a whole is not linear but rather quadratic in the sample location.

Now that we have explained the methodology of obtaining the homography and using that homography for transforming the images from image plane to world plane, we shall now show our results of the given tasks with brief explanation while referring to first two sections wherever necessary. But before that, we shall show the two images that are used in this homework. Please note that the points that are used for finding homographies have been shown in the start where the test images are shown.


**Figure 2:** Frame.jpg


**Figure 3:** Audrey.jpg

## Task 1

- In this task, we are supposed to project Fig. 3 (Audrey.jpg) into the frame in Fig. 2 (Frame.jpg) defined by points P, Q, R, S.
- For this first we have to find the homography between Fig. 2 (Frame.jpg) and Fig. 3 (Audrey.jpg).
- Then using that homography we have to project Fig. 3 (Audrey.jpg) into the frame defined by points P, Q, R, S.
- To find the homography, first we establish point correspondences using points P′, Q′, R′, S′ in Fig. 3 (Audrey.jpg) which correspond to points P, Q, R and S in Fig. 2 (Frame.jpg).
- After the homography is found, then we transform the image Fig. 3 (Audrey.jpg) on to the plane of Fig. 2 (Frame.jpg) using the methodology describe in first two sections.
- Following are the results for this transformation.


**Figure 4:** Result of Task 1

### Task 2

- In this task, we are supposed to project the plane of Fig. 2 (Frame.jpg) on to the plane of Fig. 3 (Audrey.jpg) such that the frame defined by A, B, C, D in Fig. 2 (Frame.jpg) fits around the face in Fig. 3 (Audrey.jpg) .
- For this first we have to find the homography between Fig. 2 (Frame.jpg) and Fig. 3 (Audrey.jpg).
- Then using that homography we have to project Fig. 2 (Frame.jpg) on to the plane of Fig. 3 (Audrey.jpg).
- To find the homography, first we establish point correspondences using points P′, Q′, R′, S′ in Fig. 3 (Audrey.jpg) which correspond to points A, B, C and D in Fig. 2 (Frame.jpg).
- After the homography is found, then we transform the image Fig. 2 (Frame.jpg) on to the plane of Fig. 3 (Audrey.jpg) using the methodology describe in first two sections.
- Following are the results for this transformation.

**Figure 5:** Result of Task 2

## Task 3

- In task 3, we have to project the resultant images of task 1 and task 2 on to the world plane for image in Fig. 2 (Frame.jpg).
- The world coordinates for Fig. 2 (Frame.jpg) are given in Fig. 6.
- For this, we first find the homography between the results of Task 1 and Task 2 using the appropriate frames i.e. for projecting the result of task 1, we use point P, Q, R, S in Fig. 6 and for establishing point correspondences and for projecting the result of task 2, we use point A, B, C, D in Fig. 6.
- Once the homography is found, then we simply project the results of task 1 and task 2 on to the world plane using the corresponding homographies. How this is done is already described in the first two introductory sections.

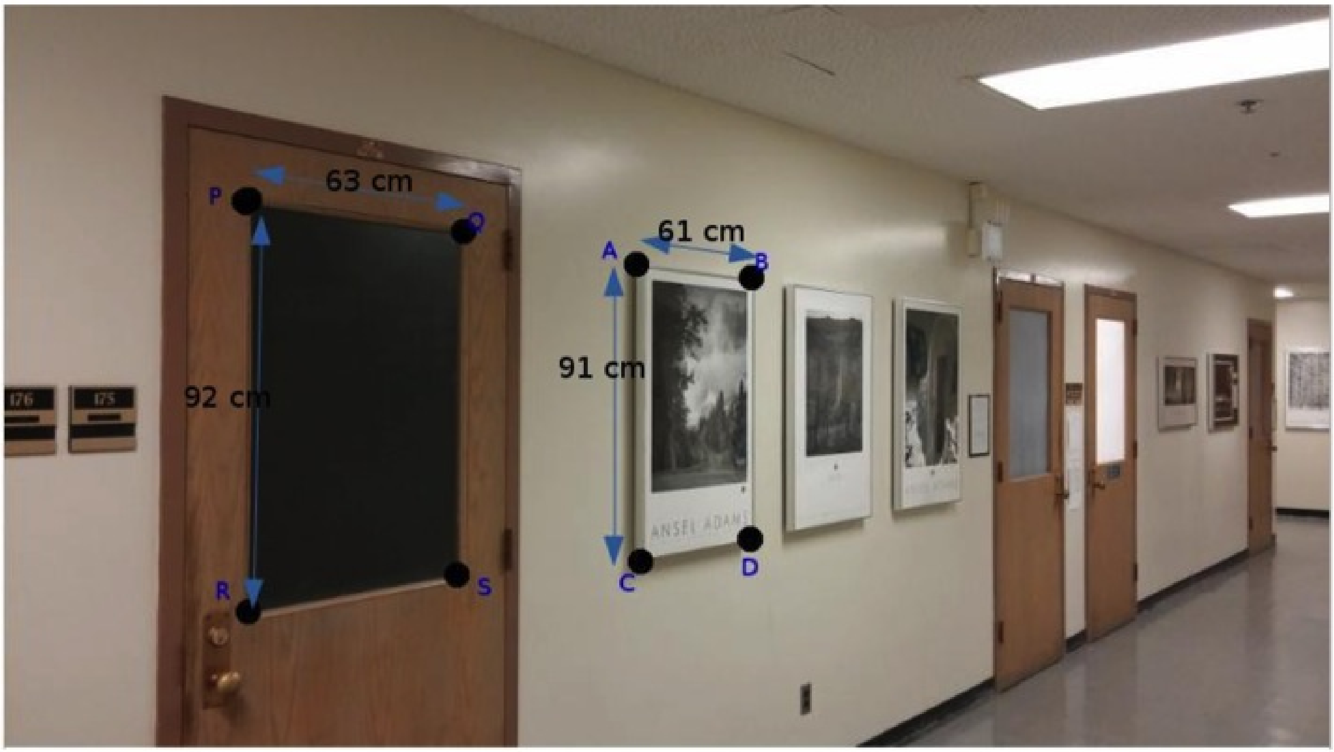


**Figure 6:** World Coordinates

Following are the results.

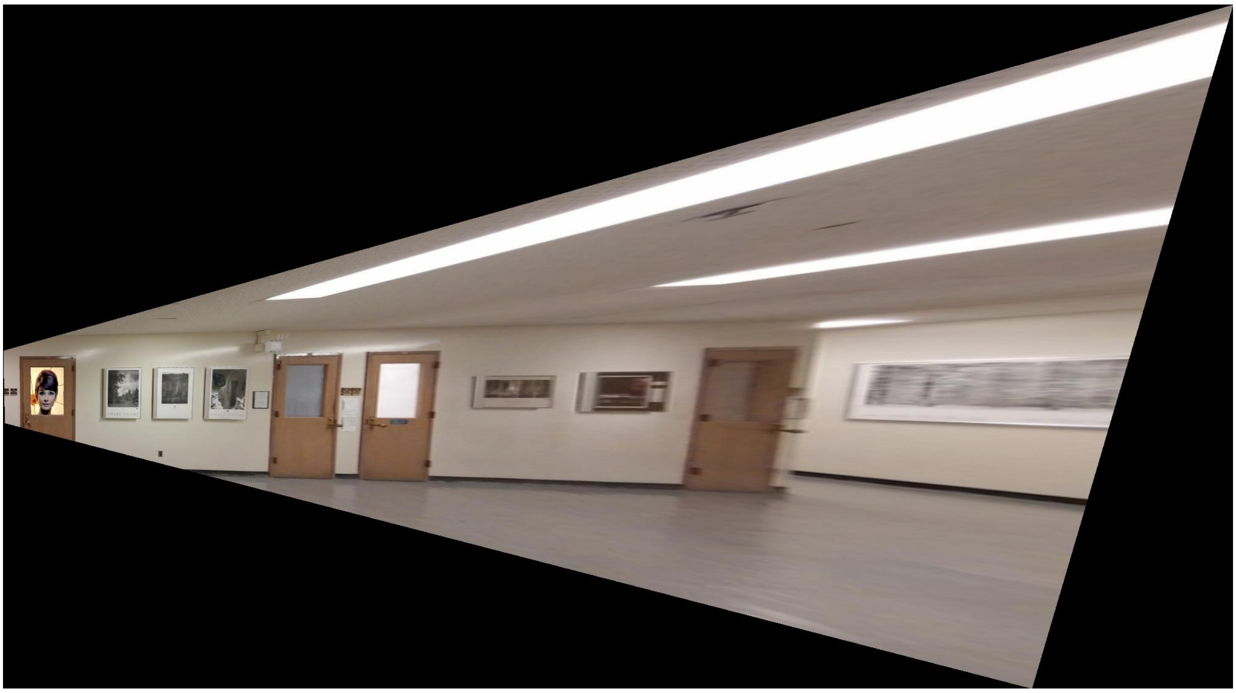

**Figure 7:** Projection of Task 1 Result on to the world plane

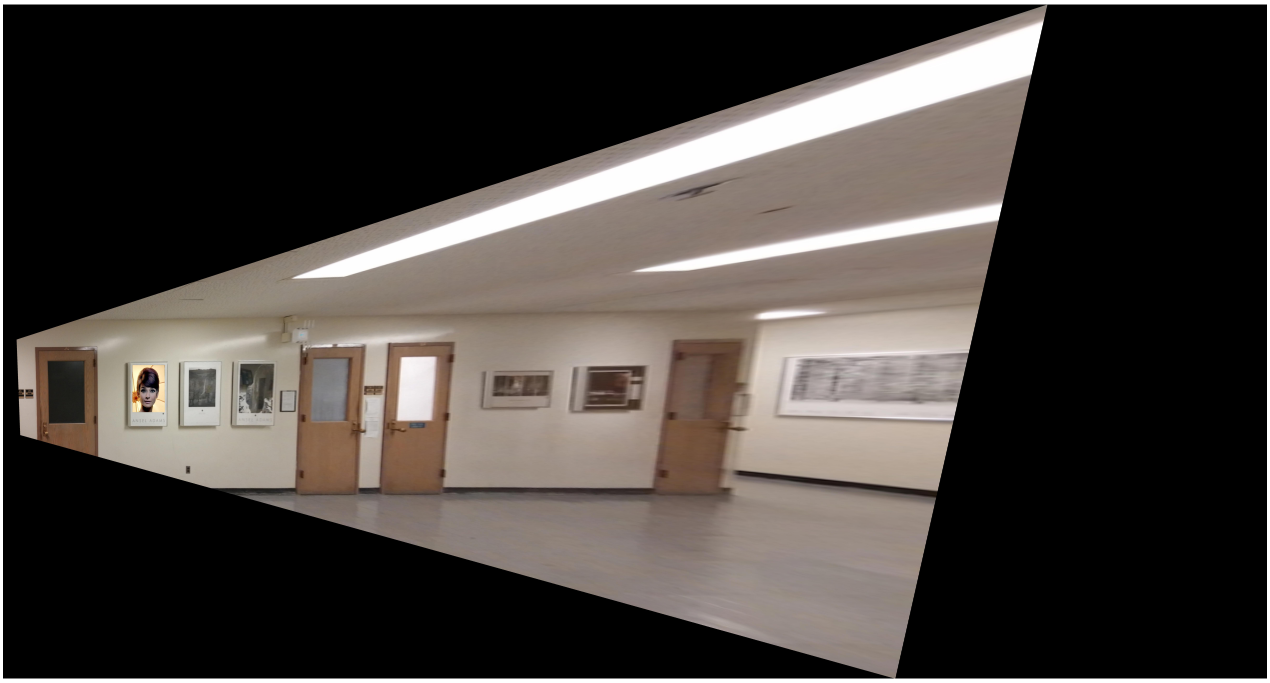

**Figure 8:** Projection of Task 2 Result on to the world plane

## Repitition of all three tasks using your own images

We have to perform the same three tasks again using our own images. Following are the images that we shall be using for this purpose. Again the points that are used for finding homographies have been shown in the start where the test images are shown. Both the frames are 102cm tall and 75cm wide.


**Figure 9:** Frame 2.jpg

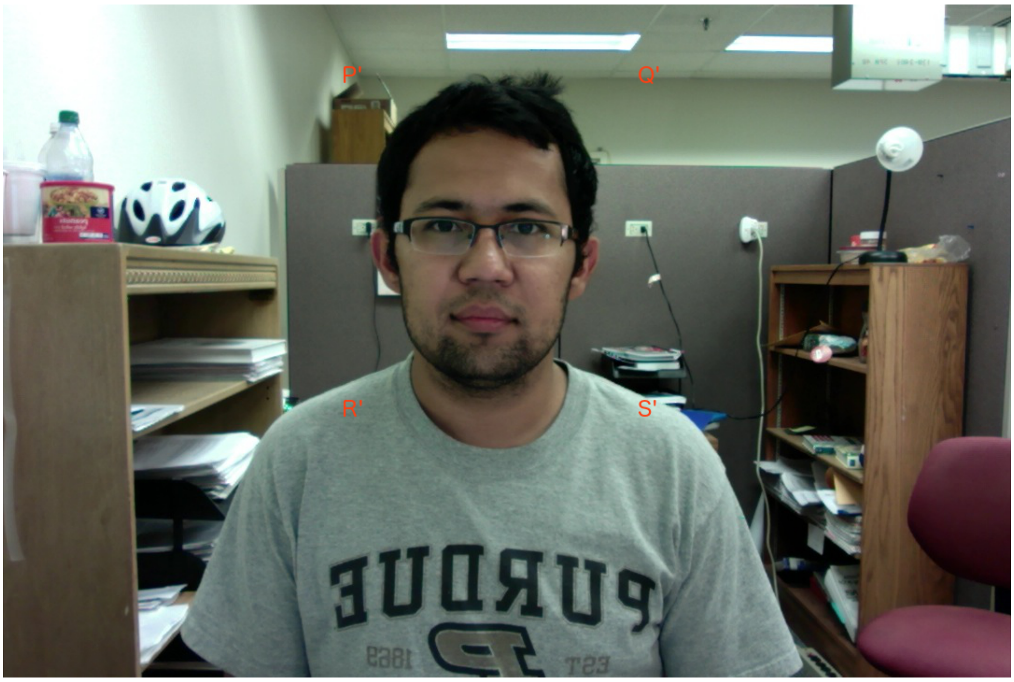


**Figure 10:** Zeeshan Nadir.jpg

**Figure 11:** Result of Task 1

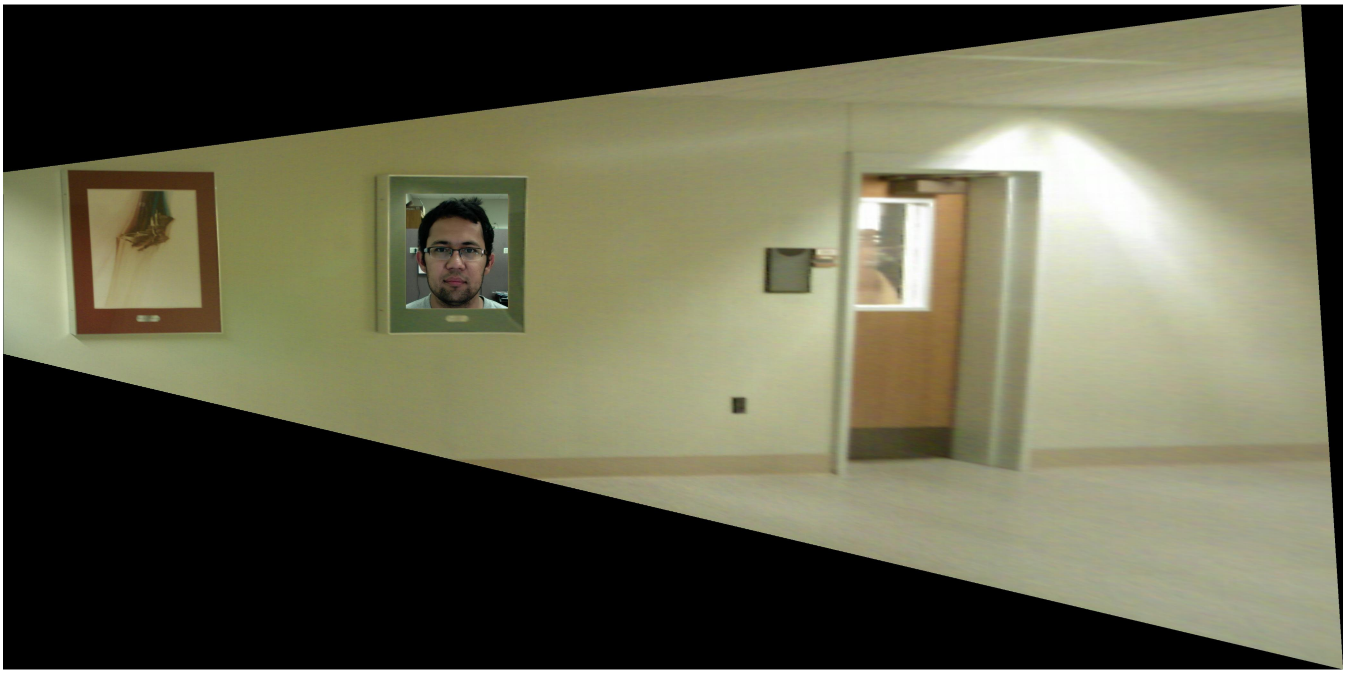

**Figure 12:** Result of Task 2

**Figure 13:** Projection of Task 1 Result on to world plane

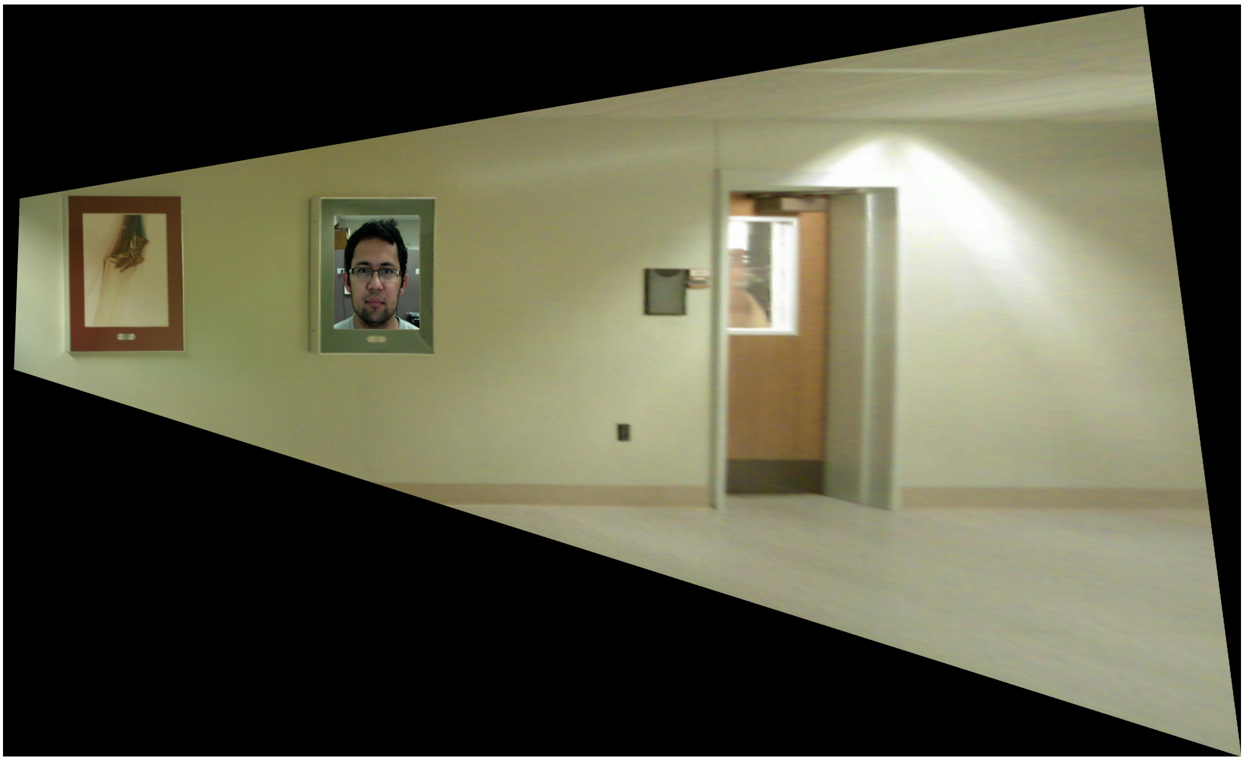

**Figure 14:** Projection of Task 2 Result on to world plane

# Source Code

Following are the source codes used for each of the 3 tasks. Please note that I am not using any openCV functions here for finding homographies etc. Except for reading and writing images, every function has been implemented.

Here is the MATLAB code for **Task 1**.

```
path_load = ['/Users/zeeshannadir/purdue/ECE661/Fall 2014/hw 2/Pics/'];

[x] = imread([path_load 'Frame.jpg']);
image(x);

[y] = imread([path_load 'Audrey.jpg']);
figure(2)
image(y);

% indicating the points in Fig. 2 (Frame.jpg) for finding point correspondences
P = [188 154];
Q = [346 177];
R = [185 462];
S = [344 433];

% indicating the points in Fig. 3 (Audrey.jpg) for drawing a box around face
P_ = [36 103];
Q_ = [305 103];
R_ = [36 439];
S_ = [305 439];

% Cropping Audrey.jpg for convenience using the four points P_ , Q_ , R_ , S_
y = y(103:439,36:305,:);

% After cropping, now indicating the corner points for establishing point
% correspondences
a_ = [1 1];
b_ = [size(y,2) 1];
c_ = [1 size(y,1) ];
d_ = [size(y,2) size(y,1)];


% establish homography
x_w = [P; Q; R; S;];
x_i = [a_; b_; c_; d_;];

% converting the images to double for calculations
x = double(x);
y = double(y);

H = findHomography (x_i,x_w);
H_ = H^(-1); % find H inverse once for all

% finding the boundries of the image in the world plane i.e. finding
% boundary of Audrey.jpg in the plane of Frame.jpg

% give physical is a function that returns physical coordinates from
```

```
% homogeneous coordinates
[i1 i2]=give_physical ( H_* [a_.';1] ); a(1) = round(i1); a(2) = round(i2);
[i1 i2]=give_physical ( H_* [b_.';1] ); b(1) = round(i1); b(2) = round(i2);
[i1 i2]=give_physical ( H_* [c_.';1] ); c(1) = round(i1); c(2) = round(i2);
[i1 i2]=give_physical ( H_* [d_.';1] ); d(1) = round(i1); d(2) = round(i2);


% initialize the output image
z = x; % output image

% compute the offsets so that only the grid points within the boundary are
% considered
tx1 = min([a(1) b(1) c(1) d(1)]);
tx2 = max([a(1) b(1) c(1) d(1)]);

ty1 = min([a(2) b(2) c(2) d(2)]);
ty2 = max([a(2) b(2) c(2) d(2)]);

% compute the height and width of projected image into the world plane
height = ty2-ty1;
width  = tx2-tx1;

tx = -tx1 + 1; % now we have the proper offsets that could be added
ty = -ty1 + 1; % now we have the proper offsets that could be added

temp_var = zeros(1,1,3); % a temporary variable for book keeping
for m = 1:1:height
    for n=1:1:width
        [i1 i2]=give_physical ( H * [n-tx;m-ty;1] );
        temp = biLinear(i1,i2,y);
        % check if bilinear didn't return zero, it may return zero if the index
        % where we want to interpolate is outside the domain of image plane
        if (any (temp ≠ temp_var))
            z(m-ty,n-tx,:) = temp; % note all three channels (rgb) are ...
                copied at once
        end
    end
end

% convert the result back into Unsigned integer
z=uint8(z);
figure;
imshow(z);
imwrite(z,[path_load 'task1_result.tif']);
```

Here is the MATLAB code for **Task 2**.

```
close all; clc; clear
% Please note that all of this work takes into account of the fact that in ...
    the formulation we write
% x coordinate before y i.e. (x,y) however in MATLAB the first index
% traverses the rows of the matrix i.e. in y direction and second index
% traverses the columns i.e. in x direction
path_load = ['/Users/zeeshannadir/purdue/ECE661/Fall 2014/hw 2/Pics/'];

[x] = imread([path_load 'Frame.jpg']);
image(x);

[y] = imread([path_load 'Audrey.jpg']);
figure(2)
image(y);

% indicating the points in Fig. 2 (Frame.jpg) for finding point correspondences
P = [491 211];
Q = [562 220];
R = [490 372];
S = [563 364];

% indicating the corner points for establishing point correspondences
P_ = [36 103];
Q_ = [305 103];
R_ = [36 439];
S_ = [305 439];

% establish homography by providing the 4 point correspondences
x_w = [P; Q; R; S;];
x_i = [P_; Q_; R_; S_;];

% converting the images to double for calculations
x = double(x);
y = double(y);

H = findHomography (x_i,x_w);
H_ = H^(-1); % find H inverse once for all


% now we find the width and height of the projected image by finding the
% corner points in the image world

% these are the corner points of Frame.jpg for which we shall find boundary
% in the plane of Audrey.jpg
a = [1 1];
b = [size(x,2) 1];
c = [1 size(x,1) ];
d = [size(x,2) size(x,1)];

% give physical is a function that returns physical coordinates from
% homogeneous coordinates
[i1 i2]=give_physical ( H* [a.';1] ); a_(1) = round(i1); a_(2) = round(i2);
[i1 i2]=give_physical ( H* [b.';1] ); b_(1) = round(i1); b_(2) = round(i2);
[i1 i2]=give_physical ( H* [c.';1] ); c_(1) = round(i1); c_(2) = round(i2);
[i1 i2]=give_physical ( H* [d.';1] ); d_(1) = round(i1); d_(2) = round(i2);

% compute the offsets so that only the grid points within the boundary are
```

```
% considered

tx1 = min([a_(1) b_(1) c_(1) d_(1)]);
tx2 = max([a_(1) b_(1) c_(1) d_(1)]);

ty1 = min([a_(2) b_(2) c_(2) d_(2)]);
ty2 = max([a_(2) b_(2) c_(2) d_(2)]);

height = ty2-ty1;
width  = tx2-tx1;

tx = -tx1 + 1; % now we have the proper offsets that could be added
ty = -ty1 + 1; % now we have the proper offsets that could be added


z = zeros(height,width,3); % initialize output image

for m = 1:1:height
    for n=1:1:width
        [i1 i2]=give_physical ( H_ * [n-tx;m-ty;1] ); % note all three ...
            channels (rgb) are copied at once
        z(m,n,:) = biLinear(i1,i2,x);
    end
end

% Once Frame.jpg is projected in the plane of Audrey.jpg, we can paste the
% picture of Audrey.jpg inside the frame since both are in same planes now

for m=103:1:439 % traverse in y direction
    for n=36:1:305 % traverse in x direction
        z(m+ty,n+tx,:) = y(m,n,:);
    end
end

% convert the result back into unsigned integer
z=uint8(z);
figure;
imshow(z);
imwrite(z,[path_load 'task2_result.tif']);
```

Here is the MATLAB code for **Task 3**.

```
path_load = ['/Users/zeeshannadir/purdue/ECE661/Fall 2014/hw 2/Pics/'];

[x] = imread([path_load 'Frame.jpg']);
image(x);

[y] = imread([path_load 'Audrey.jpg']);
figure(2)
image(y);

figure
[T1_result] = imread([path_load 'task1_result.tif']);
image(T1_result);

figure
[T2_result] = imread([path_load 'task2_result.tif']);
image(T2_result);

% indicating the points in Fig. 2 (Frame.jpg) for finding point correspondences
P_ = [491 211];
Q_ = [562 220];
R_ = [490 372];
S_ = [563 364];

% length and width (in cm) of glass frame in the door in the world plane
frame_length = 92;
frame_width  = 63;

% providing the world coordinates of the glass frame in the door in world
% plane
P = [0 0];
Q = [frame_width 0];
R = [0 frame_length];
S = [frame_width frame_length];

% Finding the Homogrphy by providing the 4 point correspondences
x_w = [P; Q; R; S];
x_i = [P_; Q_; R_; S_];

x = double(x);
y = double(y);

H = findHomography (x_i,x_w);
H_ = H^(-1); % find H inverse once for all

% now we find the width and height of the projected image by using  the
% corner points in the image world

% these are the corner points of Result of Task 1 for which we shall find
% boundary in the world plane

% give physical is a function that returns physical coordinates from
% homogeneous coordinates
a_ = [1 1];
b_ = [size(T2_result,2) 1];
c_ = [1 size(T2_result,1)];
d_ = [size(T2_result,2) size(T2_result,1)];
```

```
[i1 i2]=give_physical ( H_* [a_.';1] ); a(1) = i1; a(2) = i2;
[i1 i2]=give_physical ( H_* [b_.';1] ); b(1) = i1; b(2) = i2;
[i1 i2]=give_physical ( H_* [c_.';1] ); c(1) = i1; c(2) = i2;
[i1 i2]=give_physical ( H_* [d_.';1] ); d(1) = i1; d(2) = i2;

% finding the total height and width (in cm) of image in the world plane
height_cm = max([c(2)-a(2) d(2)-b(2)]); % compute height in cm
width_cm = max([b(1)-a(1) d(1)-c(1)]); % compute width in cm

% Finding the aspect ratio
aspect_ratio = width_cm/height_cm;

% set the width of output the same as input and select height according to
% aspect ratio
width = size(T2_result,2);
height = round ( (aspect_ratio^-1) * width);

pixel_height = height_cm/height;
pixel_width = width_cm/width;

z = zeros(height,width,3) ; % initialize output image

% compute the offsets so that only the grid points within the boundary are
% considered

tx = min([a(1) b(1) c(1) d(1)]);
ty = min([a(2) b(2) c(2) d(2)]);

tx = tx/pixel_width;
ty = ty/pixel_height;

tx = -tx + 1; % now we have the proper offsets that could be added
ty = -ty + 1; % now we have the proper offsets that could be added

for m = 1:1:height
    for n=1:1:width
        [i1 i2]=give_physical ( H * ...
            [(n-tx).*pixel_width;(m-ty).*pixel_height;1] );
         z(m,n,:) = biLinear(i1,i2,T2_result); % note all three channels ...
             (rgb) are copied at once
    end
end

% converting the result back to unsigned integer
z=uint8(z);
figure;
image(z);
imwrite(z,[path_load 'task3_2_result.tif']);
```

Here is the MATLAB code for function **findHomography**.

```
function [H] = findHomography(x_i,x_w)
% The rows of x_i are coordinates of points of scene in image plane
% The rows of x_w are coordinates of points of scene in real world plane
if (size(x_i,1) ≠ size(x_w,1))
    disp(' There should be equal no. of points in x_i and x_w');
    return;
end
N = size(x_i,1);
A = zeros (2*N,8); % forming the system matrix
b = zeros(2*N,1);    % forming the RHS of equation

% Filling up the matrix of coefficients
for k=1:1:N
    A(2*k-1,:)=[x_w(k,1) x_w(k,2) 1 0 0 0 -x_i(k,1)*x_w(k,1) ...
        -x_i(k,1)*x_w(k,2)];
    A(2*k,:)=[0 0 0 x_w(k,1) x_w(k,2) 1 -x_i(k,2)*x_w(k,1) -x_i(k,2)*x_w(k,2)];
end

% Filling up the vector of constants
for k=1:1:N
    b(2*k-1) = x_i(k,1);
    b(2*k) = x_i(k,2);
end
% Finding the lest squares estimate
% (its better to use this formula just in case we provide more than 4 points ...
    to the function)
h = (A.' * A)^-1 * ( A.' * b );
% Assigning the values to H matrix
H = zeros(3,3);
H(1,1) = h(1);
H(1,2) = h(2);
H(1,3) = h(3);
H(2,1) = h(4);
H(2,2) = h(5);
H(2,3) = h(6);
H(3,1) = h(7);
H(3,2) = h(8);
H(3,3) = 1;
end
```

Here is the MATLAB code for function **give_phsical**.

```
% This function accepts the homogeneous coordinates of a ponit
% and returns the physical coodinates of the same point
function [x1,x2] = give_physical (x)
x1 = x(1)/x(3);
x2 = x(2)/x(3);
end
```

Here is the MATLAB code for function **biLinear**.

```
function [f]= biLinear (x,y,Z)
% Z is the image where you get the values from
% x , y is the location which would be given a value from Z through
% interpolation

% first check if we are at a valid coordinate
S = size(Z);
if ((y<1) || (y>S(1)) )
    f=zeros(1,1,3);
elseif ((x<1) || (x>S(2)) )
    f = zeros(1,1,3);
else
    x1 = floor(x);
    x2 = ceil(x);
    y1 = floor(y);
    y2 = ceil(y);
    f = ( (x2-x)*(y2-y) ) / ( (x2-x1)*(y2-y1) ) * Z(y1,x1,:) + ...
        ( (x-x1)*(y2-y) ) / ( (x2-x1)*(y2-y1) ) * Z(y1,x2,:) + ...
        ( (x2-x)*(y-y1) ) / ( (x2-x1)*(y2-y1) ) * Z(y2,x1,:) + ...
        ( (x-x1)*(y-y1) ) / ( (x2-x1)*(y2-y1) ) * Z(y2,x2,:);
end
end
```