CS 532 3D Computer Vision, Homework 1

PART 1) Warping

In this part of the Homework, we have to perform warping of the given image in such a way that the top view of the basket ball court is visible from an image where the orientation of the court is different. To perform this, I first create the Homography matrix, by taking the points on the original image having correspondences on the new image's corners(940 by 500). The eight points taken on the original image and the new image are, (245,45) -> (1,1), (420,70) -> (1,500), (1,195) -> (940,1) and (300,295) ->(940,500). Then, using inverse warping technique, matrix multiplication of inverse of the Homography matrix was done with the homogeneous co-ordinates of the new image, converting the co-ordinates ranging from (1,1,1) to (940,500,1), to a given set of new co-ordinates(normalized) which could be found on the original image. Using these co-ordinates, the RGB values at these co-ordinates were then transferred to the corresponding co-ordinates of the new image, hence creating a warping. If the co-ordinates obtained by the matrix multiplication(H^-1 * (x_new,y_new,1)) were integers, then the RGB values were directly transferred to the new image co-ordinates but if they lied between some pixel co-ordinates say for example (240.36, 200.67), then Bilinear Interpolation for RGB values at these places were performed. The Bilinear Interpolation takes in account the pixels around the floating point pixel obtained and performs a mathematical operation for finding out the weight of the pixel value nearest to it. The formula for it is given as,

f(x,y) = (1-a)(1-b)*f(i,j) + a(1-b)*f(i+1,j) + ab*f(i+1, j+1) + (1-a)b*f(i, j+1)

Here, a = 0.36, b = 0.67 and f(i,j), f(i+1,j), f(i+1, j+1) , f(i, j+1) are the RGB values at pixel position (i,j), (i+1,j), (i+1, j+1) , (i, j+1) respectively. Hence, filling up the places where floating point co-ordinates appear.

CODE:

```
clc
clear all;
close all;

I   =   imread('C:\Users\Jolton\Desktop\Files\Homeworks\Semester3\3D  Computer
Vision\Homework1\basketball-court.ppm');
I_warped = zeros(940,500,3);
% I_warped(:,:,1) = 255; I_warped(:,:,2) = 255; I_warped(:,:,3) = 0;

[rows,cols,color] = size(I);
figure
imshow(I);

%% Finding the Homography matrix

x1 = 245; y1 = 45;
x2 = 420; y2 = 70;
x3 = 1; y3 = 195;
x4 = 300; y4 = 295;

x11 = 1; y11 = 1;
x22 = 1; y22 = 500;
```

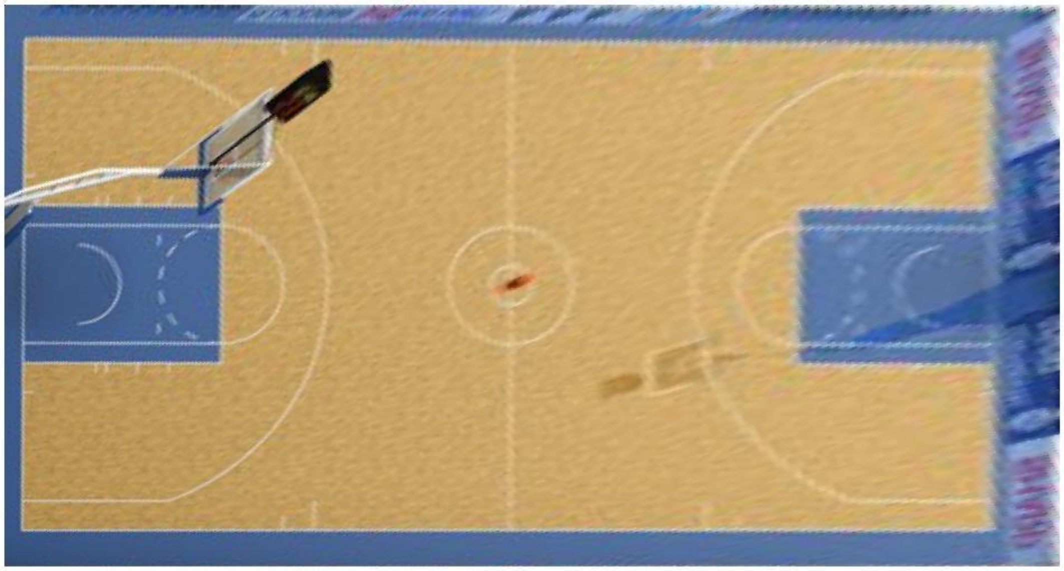
```
x33 = 940; y33 = 1;
x44 = 940; y44 = 500;
%
% x1 = 0; y1 = 194;
% x2 = 245 ; y2 = 45;
% x3 = 418; y3 = 70;
% x4 = 296; y4 = 300;

% x11 = 0; y11 = 0;
% x22 = 939 ; y22 = 0;
% x33 = 939; y33 = 499;
% x44 = 0; y44 = 499;

p1_prime = [-x1 -y1 -1 0 0 0 x1*x11 y1*x11 x11; 0 0 0 -x1 -y1 -1 x1*y11 y1*y11
y11];
p2_prime = [-x2 -y2 -1 0 0 0 x2*x22 y2*x22 x22; 0 0 0 -x2 -y2 -1 x2*y22 y2*y22
y22];
p3_prime = [-x3 -y3 -1 0 0 0 x3*x33 y3*x33 x33; 0 0 0 -x3 -y3 -1 x3*y33 y3*y33
y33];
p4_prime = [-x4 -y4 -1 0 0 0 x4*x44 y4*x44 x44; 0 0 0 -x4 -y4 -1 x4*y44 y4*y44
y44];

P = [p1_prime; p2_prime; p3_prime; p4_prime];

[U,S,V] = svd(P);
%
H = V(:,end); %Homography Matrix

H = vec2mat(H,3);

H_inverse = inv(H);
H_inv = H_inverse;

X = zeros(3,1);
X_new_cord = zeros(1,2);

a = 0;
b = 0;

%
for i = 1:size(I_warped,2)
    for j = 1:size(I_warped,1)
% for i = 1
%     for j = 2
            X =  H_inv*[j;i;1];



            X_new_cord = [X(1,1)/X(3,1), X(2,1)/X(3,1)];

            % Bilinear Interpolation
            R = (X_new_cord - floor(X_new_cord));
                if uint8(R(1)) == 1 && uint8(R(2)) == 1
%                      a = 0; b = 0;
```

```
                     I_warped(j,i,1)                                        =
I(uint32(X_new_cord(2)),uint32(X_new_cord(1)),1);
                     I_warped(j,i,2)                                        =
I(uint32(X_new_cord(2)),uint32(X_new_cord(1)),2);
                     I_warped(j,i,3)                                        =
I(uint32(X_new_cord(2)),uint32(X_new_cord(1)),3);

                else
                    a = R(1); b = R(2);
                    if floor(X_new_cord(1)) < 1 || floor(X_new_cord(1)) > cols
|| floor(X_new_cord(2)) < 1 || floor(X_new_cord(2)) > rows
                        continue;
                    else
                     R11  =  I(  floor(X_new_cord(2)),floor(X_new_cord(1)),1);
R12 = I( floor(X_new_cord(2))+1,floor(X_new_cord(1)),1);
                     R22                            =                       I(
floor(X_new_cord(2))+1,floor(X_new_cord(1))+1,1);       R21        =        I(
floor(X_new_cord(2)),floor(X_new_cord(1))+1,1);

                     G11  =  I(  floor(X_new_cord(2)),floor(X_new_cord(1)),2);
G12 = I( floor(X_new_cord(2))+1,floor(X_new_cord(1)),2);
                     G22                            =                       I(
floor(X_new_cord(2))+1,floor(X_new_cord(1))+1,2);       G21        =        I(
floor(X_new_cord(2)),floor(X_new_cord(1))+1,2);

                     B11  =  I(  floor(X_new_cord(2)),floor(X_new_cord(1)),3);
B12 = I( floor(X_new_cord(2))+1,floor(X_new_cord(1)),3);
                     B22                            =                       I(
floor(X_new_cord(2))+1,floor(X_new_cord(1))+1,3);       B21        =        I(
floor(X_new_cord(2)),floor(X_new_cord(1))+1,3);

                     I_warped(j,i,1)  =  round((1-a)*(1-b)*R11  +  a*(1-b)*R21  +
a*b*R22 + b*(1-a)*R12);
                     I_warped(j,i,2)  =  round((1-a)*(1-b)*G11  +  a*(1-b)*G21  +
a*b*G22 + b*(1-a)*G12);
                     I_warped(j,i,3)  =  round((1-a)*(1-b)*B11  +  a*(1-b)*B21  +
a*b*B22 + b*(1-a)*B12);
                    end
                end
    end
end
I_warped = uint8(I_warped);
% I_warped = imrotate(I_warped, -90);
figure
imshow(I_warped);
```

INPUT:

SHARP
SHARP
DODGE

OUTPUT:



PART 2) Dolly Zoom

Method 1)

In this part of the assignment we are supposed to perform the dolly zoom on the given pointCloud data. The code provided performs the pointCloud to image operation. We need to begin the process by setting the bounding box of the Foreground object to be of approximately 640x400. This is done by observing the bounding box on the image and noting the value of 't' in the code which causes it. We get the initial start value of the z-axis to be 3.8650 and we begin the process from here. The dolly zoom is an illusion caused by setting the focus on a subject in the image and considering other elements as the background. Once the focus is set, we need to move the camera ahead/behind and do a zoom-out/zoom-in as we step ahead/behind to create an illusion. The Foreground axes points are considered and used to create the illusion. The intrinsic parameters are changed in a manner in which,

$(fy*Y)/Z = 640$ should remain constant throughout the process and

$(fx*X)/Z = 400$ should remain constant throughout the process

Hence, we change the Intrinsic Matrix K and fill the values of fy and fx in K in a manner which will carry out the dolly zoom by keeping the bounding box constant.

Method 2)

The same can be obtained by doing a similar method which gives Dolly Zoom effect. In this method, the value of the initial Z value is used which gives a bounding box of 640x400 around the foreground object. The ratio of fx/Z and fy/Z is kept the same throughout the changes in Z(translation). This will help keep the foreground object the same size and create a Dolly Zoom illusion.

CODE 1:

```
% Sample use of PointCloud2Image(...)
%
% The following variables are contained in the provided data file:
%
BackgroundPointCloudRGB,ForegroundPointCloudRGB,K,crop_region,filter_size
% None of these variables needs to be modified


clc
clear all
%                                load                                variables:
BackgroundPointCloudRGB,ForegroundPointCloudRGB,K,crop_region,filter_size)
load data.mat

data3DC = {BackgroundPointCloudRGB,ForegroundPointCloudRGB};
R       = eye(3);

move      = [0  0  -0.025]';  %  Choosing  a  value  which  does  not  go  beyond  the
foreground

for step = 0:74
   tic
   fname       = sprintf('SampleOutput3%03d.jpg',step);
   display(sprintf('\nGenerating %s',fname));
   t           = step * move;
   Z      =   3.8650  +  t(3);  % Gives  a  bounding  box  for  the  foreground  as
approximately 640x400 (actual value -> 627 x 438)...
                          % 3.8650 is found out approximately by finding
                          % the value of t which approximately bounds the
                          % foreground to the bounding box

   K(1,1)          = (640/(0.0529 + 0.5)) * Z; % fy

   K(2,2)       = (400/(0.3287 + 0.0589)) * Z;  % fx
   M           = K*[R t];
   im          = PointCloud2Image1(M,data3DC,crop_region,filter_size);
   imwrite(im,fname);
   toc
end
```

CODE 2)

```
% Sample use of PointCloud2Image(...)
%
% The following variables are contained in the provided data file:
%
BackgroundPointCloudRGB,ForegroundPointCloudRGB,K,crop_region,filter_size
% None of these variables needs to be modified


clc
clear all
% load variables:
BackgroundPointCloudRGB,ForegroundPointCloudRGB,K,crop_region,filter_size)
load data.mat

data3DC = {BackgroundPointCloudRGB,ForegroundPointCloudRGB};
R       = eye(3);

move    = [0 0 -0.025]'; % Choosing a value which does not go beyond the
foreground

for step = 0:74
   tic
   fname       = sprintf('SampleOutput3%03d.jpg',step);
   display(sprintf('\nGenerating %s',fname));
   t           = step * move;
   Z    =  3.8650 + t(3); % Gives a bounding box for the foreground as
approximately 640x400 (actual value -> 627 x 438)...
                          % 3.8650 is found out approximately by finding
                          % the value of t which approximately bounds the
                          % foreground to the bounding box
   ratio_x = 2759.5/3.8650;
   ratio_y = 2764.2/3.8650;
   K(1,1) = ratio_y * Z;
   K(2,2) = ratio_x * Z;
%    K(1,1)          = (640/(0.0529 + 0.5)) * Z; % fy
%    K(2,2)       = (400/(0.3287 + 0.0589)) * Z;  % fx
   M            = K*[R t];
   im           = PointCloud2Image1(M,data3DC,crop_region,filter_size);
   imwrite(im,fname);
   toc
end
```

OUTPUT:

Attached in the file on canvas with images.