Kunal Dhaimade – CS532 Homework Assignment 1

Part 1: Image Warping:

Source Code:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 14 18:59:10 2017

@author: Kunal
"""

from PIL import Image
import numpy as np

def dlt_bilest(og_im, op_im):
    og_im_ar = np.array(og_im)
    row, col, x = og_im_ar.shape
    x1, y1, x2, y2, x3, y3, x4, y4 = 0, 194, 245, 45, 418, 70, 296, 300
    x1d, y1d, x2d, y2d, x3d, y3d, x4d, y4d = 0, 0, 939, 0, 939, 499, 0, 499
    a = [[-x1, -y1, -1, 0, 0, 0, x1*x1d, y1*x1d, x1d],
         [0, 0, 0, -x1, -y1, -1, x1*y1d, y1*y1d, y1d],
         [-x2, -y2, -1, 0, 0, 0, x2*x2d, y2*x2d, x2d],
         [0, 0, 0, -x2, -y2, -1, x2*y2d, y2*y2d, y2d],
         [-x3, -y3, -1, 0, 0, 0, x3*x3d, y3*x3d, x3d],
         [0, 0, 0, -x3, -y3, -1, x3*y3d, y3*y3d, y3d],
         [-x4, -y4, -1, 0, 0, 0, x4*x4d, y4*x4d, x4d],
         [0, 0, 0, -x4, -y4, -1, x4*y4d, y4*y4d, y4d]]
    a = np.asarray(a)
    a = a.astype(np.int64)
    u, s, v = np.linalg.svd(a, full_matrices=True, compute_uv=True)
    h = np.reshape(v[8,:], (3,3))
    hi = np.linalg.inv(h)
    op_im_ar = np.array(op_im)
    o_row, o_col, x = op_im_ar.shape
    for i in range(o_row):
        for j in range(o_col):
            op_p = [[j], [i], [1]]
            or_p = np.matmul(hi, op_p)
            ox = or_p[0][0]/or_p[2][0]
            oy = or_p[1][0]/or_p[2][0]
            oxt = str(ox).split('.')
            oyt = str(oy).split('.')
            if(len(oxt[1]) <= 1 or len(oyt[1]) <= 1):
                op_im_ar[i][j] = og_im_ar[int(oy)][int(ox)]
            else:
                pt_i = int(oyt[0])
                pt_j = int(oxt[0])
```

```
            a = float('0.'+oxt[1])
            b = float('0.'+oyt[1])
            if((pt_i+1) < row and (pt_j-1) >= 0):
                red = (1-a)*(1-b)*og_im_ar[pt_i][pt_j][0] + (a)*(1-b)*og_im_ar[pt_i+1][pt_j][0] + (a)*(b)*og_im_ar[pt_i+1][pt_j-1][0] + (1-a)*(b)*og_im_ar[pt_i][pt_j-1][0]
                green = (1-a)*(1-b)*og_im_ar[pt_i][pt_j][1] + (a)*(1-b)*og_im_ar[pt_i+1][pt_j][1] + (a)*(b)*og_im_ar[pt_i+1][pt_j-1][1] + (1-a)*(b)*og_im_ar[pt_i][pt_j-1][1]
                blue = (1-a)*(1-b)*og_im_ar[pt_i][pt_j][2] + (a)*(1-b)*og_im_ar[pt_i+1][pt_j][2] + (a)*(b)*og_im_ar[pt_i+1][pt_j-1][2] + (1-a)*(b)*og_im_ar[pt_i][pt_j-1][2]
                op_im_ar[i][j] = [int(red), int(green), int(blue)]
    op_im = Image.fromarray(op_im_ar)
    return op_im

def main():
    og_im = Image.open("basketball-court.ppm")
    og_im.show()
    op_im = Image.new('RGB', (940, 500), 'white')
    op_im.show()
    op_im = dlt_bilest(og_im, op_im)
    op_im.show()
    op_im.save('basketball-court-above-dlt-bilest2.jpg')

main()
```

Output Images:

Image obtained by using Inverse Warping technique by DLT with Bilinear Interpolation.

Kunal Dhaimade – CS532 Homework Assignment 1

Image obtained by using Inverse Warping technique by DLT without Bilinear Interpolation.

Image obtained by using Inverse Warping technique by DLT with Bilinear Interpolation after interchanging a and b values.

Kunal Dhaimade – CS532 Homework Assignment 1

Part 2: Dolly Zoom:

First Method:

```
% 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]';

for step=0:74
   tic
   fname       = sprintf('CS532_HW1_Dolly_OP%03d.jpg',step);
   display(sprintf('\nGenerating %s',fname));
   t           = step * move;
   z = 3.4087 + t(3)

   fx = 400/(0.3287 + 0.05) * z
   fy = 640/(0.0529 + 0.55) * z
   K(2,2) = fx
   K(1,1) = fy

   M           = K*[R t];
   im          = PointCloud2Image(M,data3DC,crop_region,filter_size);
   imwrite(im,fname);
   toc
end
```

Second Method:

```
% 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
```

Kunal Dhaimade – CS532 Homework Assignment 1

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

data3DC = {BackgroundPointCloudRGB,ForegroundPointCloudRGB};
R       = eye(3);
move    = [0 0 -0.02]';

init_z = 3.4087;
rx = init_z/K(2,2);
ry = init_z/K(1,1);

for step=0:74
   tic
   fname       = sprintf('CS532_HW1_Dolly_OP%03d.jpg',step);
   display(sprintf('\nGenerating %s',fname));
   t           = step * move;
   z = 3.4087 + t(3);

   fx = z/rx;
   fy = z/ry;
   K(2,2) = fx;
   K(1,1) = fy;

   M           = K*[R t];
   im          = PointCloud2Image(M,data3DC,crop_region,filter_size);
   imwrite(im,fname);
   toc
end
```

Video Generation File:

```
writerObj = VideoWriter('Video.avi');
 writerObj.FrameRate = 15;
 open(writerObj);

 sourceFile = dir('D:\Study\MS\3rd Term - Fall 2017\3D Computer
Vision\Homework\1\Dolly_Data_Code\Dolly_Data_Code\OP\1\*.jpg');
 for i = 1: length(sourceFile)
  filename = strcat('D:\Study\MS\3rd Term - Fall 2017\3D Computer
Vision\Homework\1\Dolly_Data_Code\Dolly_Data_Code\OP\1\',sourceFile(i).name);
  images = imread(filename);
  frame = im2frame(images);
  writeVideo(writerObj, frame);
 end

 close(writerObj);
```

Kunal Dhaimade – CS532 Homework Assignment 1

Notes:

For Image Warping task, I used the Direct Linear Transform (DLT) method, to obtain the homography matrix and also its inverse. The points I used in the original image are $(0, 194)$, $(245, 45)$, $(418, 70)$ and $(296, 300)$, mapped onto the output image points $(0, 0)$, $(939, 0)$, $(939, 499)$ and $(0, 499)$ respectively. I used Inverse warping to fill the pixel colors, using Bilinear Interpolation to fill the pixels lying in between. I also tested by avoiding the interpolation technique by rounding off the coordinates obtained by inverse mapping to integers. I got similar results by both techniques.

For Dolly Zoom, I used 2 different techniques. In one, I used the relation given in the slides $y' = \frac{fy}{z}$, and in other, I formed the relation between the depth and the focal length myself. The first method seems to provide a better dolly zoom effect. Both the videos for Dolly Zoom have been attached with the submission folder.