2nd Assignment 776: “Computer Vision”, Fall 2016

Due date: Sept 27, 11:59 pm (by e-mail to jmf@cs.unc.edu)

Data: http://www.cs.unc.edu/~jmf/teaching/fall2016/Assignment_2.zip

What should you turn in:

1) A pdf file with the discussion of your solutions.

2) A link to the generated video sequence.

3) The matlab code used to generate the results.

Summary

The goal of this assignment is for you to apply your knowledge of the pinhole camera model by controlling both the internal and external parameters of a virtual camera in order to simulate the effect of a "dolly zoom" as well as the effects of lens radial distortion.

Provided are a Matlab function and the 3D data for generating a synthetic image corresponding to a given projection matrix. You shall use the provided code to generate an image sequence where the "dolly zoom" effect is illustrated and write your own post-processing Matlab module for radial distortion of the generated images. Aggregate the generated images into a single video in (.wmv) file format (75 frames @ 15 Hz).

Dolly zoom

The dolly zoom is an optical effect used by cinematographers. The effect consists in adjusting the distance of the camera to a foreground object in the scene, while simultaneously controlling the camera's field of view (a function of the focal length), in order for the foreground object to retain a constant size in the image throughout the entire capture sequence.

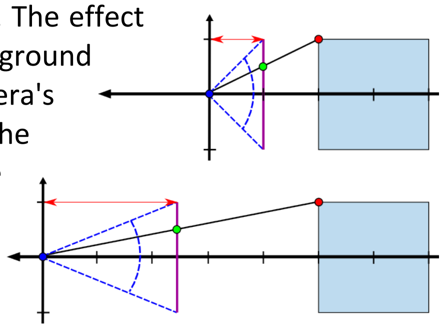
Figure 1: The dolly zoom effect

In order to approximate a photorealistic effect, you are provided a dense point cloud augmented with RGB color information. To obtain a rendered image you can use the provided rendering function ***PointCloud2Image***, which takes as input a projection matrix and transforms the 3D point cloud into a 2D image (see below for details). Your task will be to:

1) Design a sequence of projection matrices corresponding to each frame of a "dolly zoom" capture sequence and effect.

2) Use the provided code to render each of the individual images (capture frames).

Setup: Start the sequence using the camera's original internal calibration matrix K (provided in the data.mat file) and position the camera in such a way that the foreground object occupies in the initial image a bounding box of approx 400 by 640 pixels (width and height) respectively. (Per reference, positioning the camera at the origin renders the foreground object within a bounding box of size 250 by 400 pixels).

Radial Distortion

Radial distortion is the most significant type of geometric distortion in today's cameras. It is most evident in images produced with low-cost, wide-angle lenses. Radial distortion bends straight lines into circular arcs, violating the main invariance preserved in the pinhole camera model that straight lines in the world map to straight lines in the image plane. Radial distortion may appear as barrel distortion, usually arising at short focal lengths, or pincushion distortion, usually arising at longer focal lengths.

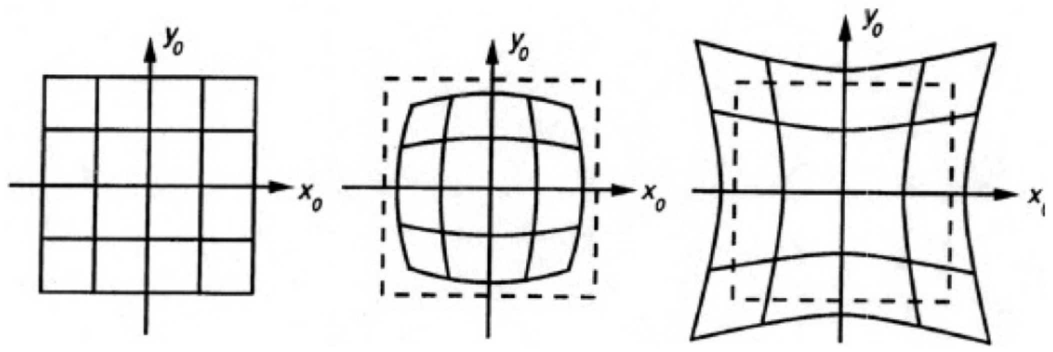

Figure 2 Barrel distortion (middle). Pincushion distortion (right)

Distortion can be compensated mathematically, first by applying a parametric distortion model, estimating the distortion coefficients and then correcting the distortion. The Brown distortion model is an example of a polynomial approximation model:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = (1 + K_1 r^2 + K_2 r^4 + K_3 r^6) \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 2K_4 uv + 2K_5 (r^2 + 2u^2) \\ K_4 (r^2 + 2v^2) + 2K_5 uv \end{bmatrix}$$

Where $(u,v)^T$ and $(u',v')^T$ are normalized camera coordinates (not image coordinates!), r = $|(u,v)^T|_2$ and K_i denotes a real valued distortion coefficient. Note that the Brown model serves to describe the pixel displacement field between a pair of images, where one of them is denoted (by convention) as "distorted", while the other as "undistorted".

Your task is to apply a "radial distortion" procedure to each of the synthetic frames generated for the dolly zoom sequence. Consider a single radial distortion coefficient K_1. After converting the pixel coordinates of the synthetically generated image into normalized camera coordinates, they will correspond to the $(u',v')^T$ coordinates. Hence, your distortion procedure needs to estimate the coordinates $(u,v)^T = g(K_1,u',v')$. In this way, you will be solving for a "backwards" radial distortion model. Once the "distorted" image coordinates have been computed, generate a new image. In you report explain the methods used to solve the problem and discuss their possible extension to a full radial distortion model.

Setup: Let the distortion parameter K_1 be defined in the range [-1.0,1.0] and interpolate linearly between these two values along the entire dolly zoom sequence.

Implementation details & Matlab Code

The file data.mat contains the scene of interest represented as a 3D point cloud, the camera internal calibration matrix to be used along with the image rendering parameters. All these variables are to be loaded into memory and need not be modified.

The file PointCloud2Image.m contains the point cloud rendering function whose signature is {img =PointCloud2Image(P,Sets3DRGB,viewport,filter_size)}. **P denotes a 3x4 projection matrix** and should be the **only parameter you will need to vary** when calling this function, as the remaining parameters should remain constant.

A simplified example of how to use the function is included in the file SampleCameraPath.m . The provided sample code does not perform the dolly zoom effect, it only displaces the camera towards the scene. It will be your task to manipulate the camera internal and external parameters to get the desired result.

The pointcloud data is contained in two variables: BackgroundPointCloudRGB and ForegroundPointCloudRGB, each comprising of a 6xN matrix. The first three rows describe the 3D coordinates of a point while the last three contain the corresponding RGB values. You may need to examine the ForegroundPointCloudRGB to determine the required camera positions. The pointcloud was generated from a single depthmap where the foreground object was masked out and its depth reduced by half.

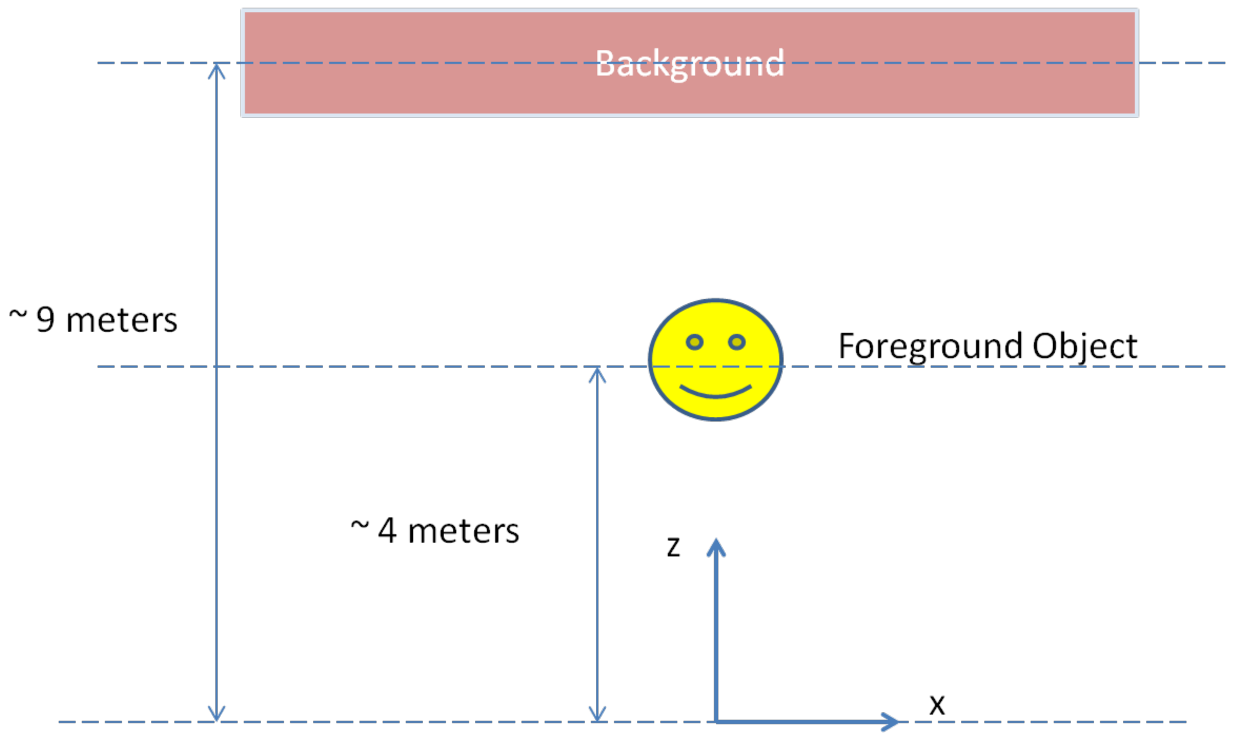

Figure 3. Birds eye view of the observed scene

The generated video should be approximately 5 seconds in length at a frame rate of 15Hz. WMV will be the only format accepted.

P.S: If you are using python for your parts of the code you can use the Matlab Python API to call the matlab functions for rendering.