CS 532: Homework Assignment 1
Due: September 21, 5:59PM

Enrique Dunn
Department of Computer Science
Stevens Institute of Technology
edunn@stevens.edu

Collaboration Policy. Homeworks will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems. Use of the Internet is allowed, but should not include searching for previous solutions or answers to the specific questions of the assignment. I will assume that, as participants in a graduate course, you will be taking the responsibility of making sure that you personally understand the solution to any work arising from collaboration.

Late Policy. No late submissions will be allowed without consent from the instructor. If urgent or unusual circumstances prohibit you from submitting a homework assignment in time, please e-mail me explaining the situation.

Submission Format. Electronic submission on Canvas is mandatory. Submit in a zip file, a pdf file:

- source code (excluding libraries),
- points used in the computation,
- resulting images,
- at most one page of text explaining anything that is not obvious. Also include the code and

output images/videos separately.

Problem 1. (60 points)

The goal is for you to apply your knowledge of Homography estimation from a set of image features in order to perform a simple image warping task. In particular, you are expected to implement

a) The DLT algorithm for homography estimation
b) 2D Bilinear interpolation to render the output image

Download the image of the basketball court from the Canvas course website. Then, generate a blank 940 × 500 image and warp the basketball court only from the source image, where it

appears distorted, to the new image so that it appears as if the new image was taken from directly above.

Notes.

- You are allowed to use image reading and writing functions, but not homography estimation or bilinear interpolations functions.

- Matlab, gimp or Irfanview (Windows only) can be used to click on pixels and record their coordinates.

Problem 2. (40 points) Dolly Zoom

The goal 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". 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.

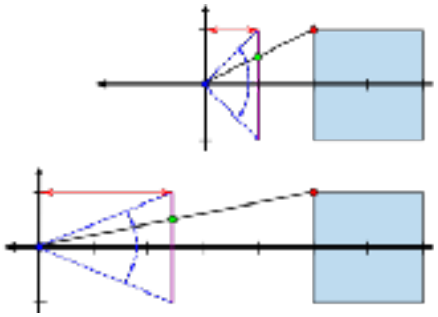

Figure 1. The Dolly Zoom Effect

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).

Notes: 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.

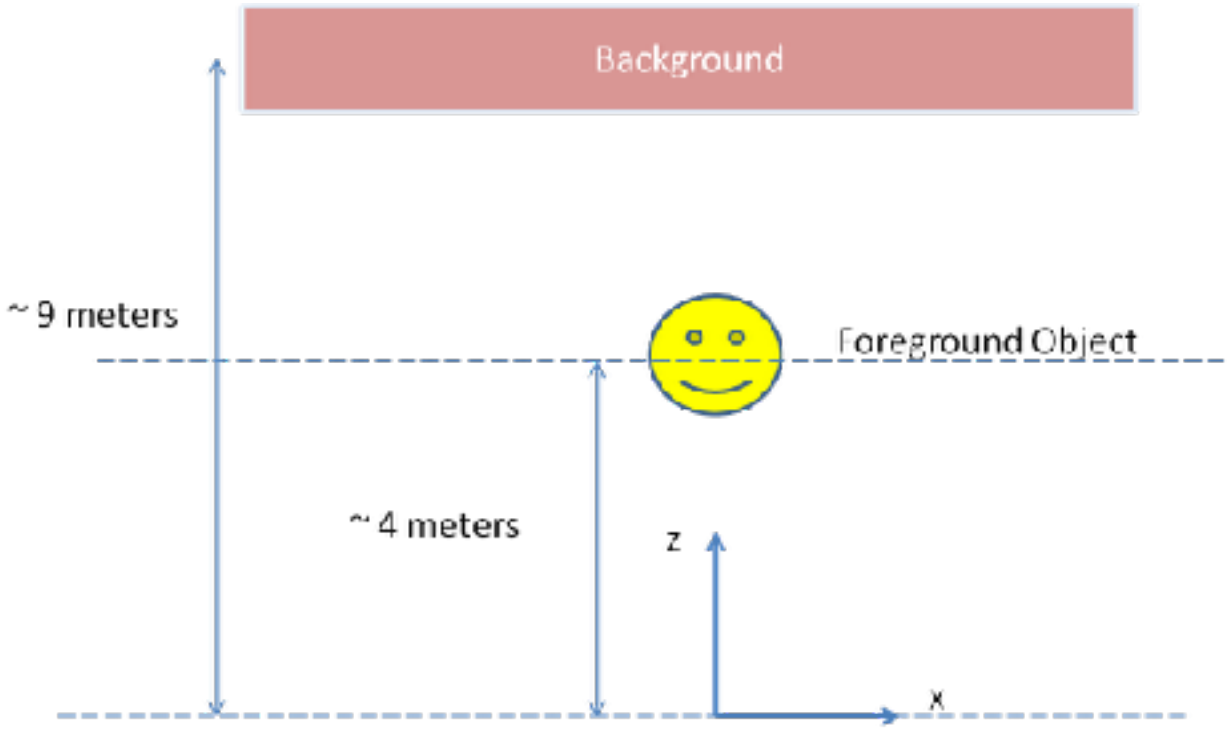

Figure 2. Birds eye view of the observed scene

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