CS 532: Homework Assignment 1
Due: February 15th, 5:59PM

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

Collaboration Policy. Homeworks may be done individually or in teams of two. It is acceptable for students of different teams 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. 3% penalty for partial 24-hour period of delay.

Submission Format. Electronic submission on Canvas is mandatory. Submit in a zip file contaning

PDF file:

- at most one page of text explaining anything that is not obvious. Also include the
- richly documented source code (excluding libraries),
- points used in the computation,
- resulting images,
- Instructions for running your code, including the execution command string that would generate your results.

Separate directory for all code

Separate directory for all generated imagery

Problem 1. (50 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 using pixel feature locations (15pts)
b) 2D Bilinear interpolation to render the output image (10 pts)
c) The DLT algorithm for homography estimation using line feature locations (25pts)

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.

- For P1a, Matlab, gimp or Irfanview (Windows only) can be used to click on pixels and record their coordinates.

- For P1c, line coordinates you are free to use the same (four) corner points used in P1a (and define lines based on their coordinates) or determine new lines (e.g. lines in the image).

Problem 2. (50 points) Object Centered motion
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 to generate a camera path that "locks-in" to foreground object (i.e. the foreground object should be and retain a constant size in the image throughout the entire capture sequence).

In order to approximate a photorealistic image generation, 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 path that performs a half circle around (i.e. centered on) the foreground object (in this case a fish statue)

2) Design a sequence of projection matrices corresponding to each frame of capture sequence

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

The main challenges are

a) Setup the camera extrinsics and intrinsics to achieve the desired initial image position
b) Design a suitable pose interpolation strategy

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 does the circling 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.

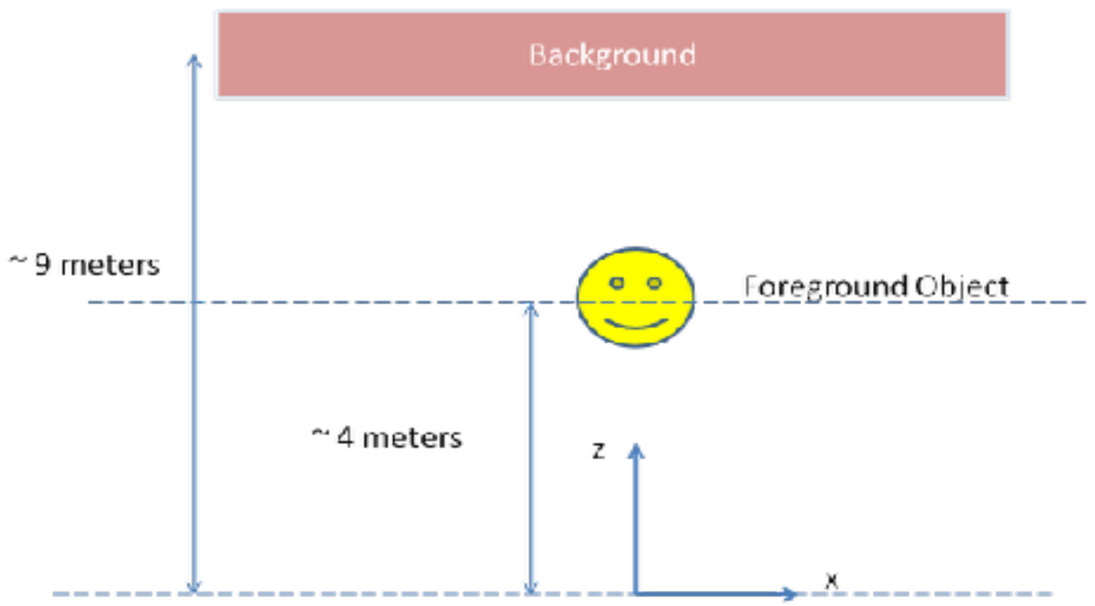

Figure 2. Birds eye view of the observed scene

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