CS410 Computer Vision

Assignment 1 - Camera Modelling, Calibration, and Estimation

Lecturer: Dr. John McDonald
Submission deadline: 03 December 2015

0. Introduction

In this assignment you will be required to write a number of python functions for (i) modelling, simulating, and visualising camera transformations, (ii) calibrating cameras from real-world data, and (iii) using plane-to-plane calibration to perform planar based augmented reality (AR).

For this assignment you should create single python file called **camera.py** containing all of the functions you develop. **Please ensure that your code is well commented.**

If you have any questions or issues during the assignment, please feel free to post them to the course forum on the moodle page.

1. Rigid body transformations

A. In this question you should write a number of functions for computing rotation matrices and homogeneous euclidean transformation matrices as specified below.

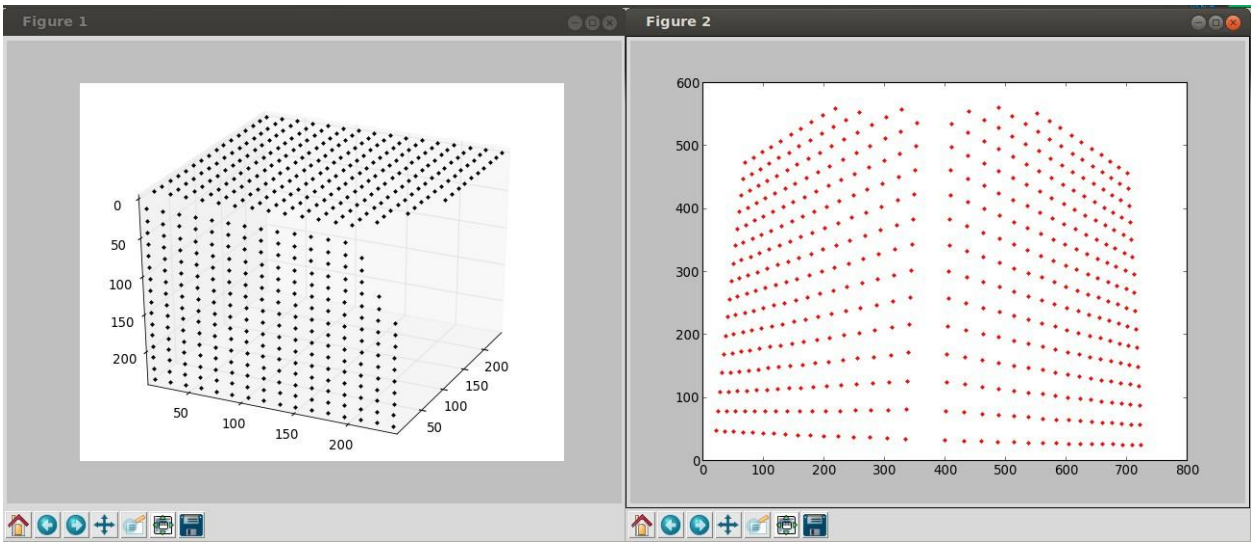
```
R = eulerToR((alpha, beta, gamma)) # compute R from Euler angles
R = expToR((w1, w2, w3)) # compute R from exponential coordinates
T = eulerToT((X, Y, Z, alpha, beta, gamma)) # compute homogeneous
transformation T from 6D pose parameters
T = expToT((X, Y, Z, w1, w2, w3)) # compute homogeneous
transformation T from 6D pose parameters
```

B. In a similar manner to the previous section, write a function called **`intrinsicToK`** that takes the intrinsic parameters of a camera and returns a perspective projection matrix for the camera.

C. Finally write a function called **`simulateCamera`** that demonstrates the combination of all the functions written so far by simulating the output of a camera raised 2 meters above a ground plane with the image plane orthogonal to the ground plane and the principal ray parallel to the ground plane and parallel to the world X axis (i.e. looking "*straight-ahead*"). As input to the camera you should create three parallel lines, each contained in the ground plane (i.e. Z=0) and parallel to the X axis (i.e. Y=-1, Y=0, and Y=1, respetively).

2. Camera calibration

Download the `data.txt` file from the course webpage. This file contains a series of world point to image point correspondences from a real-world 3D camera calibration target. The figure below shows plots of the 3D points and 2D image projections, respectively.



The `data` matrix is a N$\times$5 matrix where each row defines an individual correspondence. The first three elements of each row define the (X,Y,Z) world coordinates of the point. The fourth and fifth element of the row define the (x,y) image coordinates of the point. Write a function that takes this matrix of correspondences as input, and returns the perspective projection matrix of the camera as output.

The function should have the following prototype

```
P = calibrateCamera3D(data)
```

Provide two further functions

```
visualiseCameraCalibration3D(data, P)
evaluateCameraCalibration3D(data, P)
```

which should both take as input the original calibration data and the camera matrix P. The first function should render a single 2D plot showing (i) the measured 2D image point (i.e. as shown on the right in the above figure), and, (ii) the reprojection of the 3D calibration points as computed by P. The second function should print the mean, variance, minimum, and maximum distances in pixels between the measured and reprojected image feature locations.

3. Planar homographies

A. In this section you first will learn to use the openCV routines for calibrating a real camera (e.g. the web cam on your machine). To do this you should follow the online tutorial below: http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html
Having completed the tutorial you should be able to (i) compute the distortion (and intrinsic) parameters for your camera, and, (ii) undistort images from your camera such that the lines of the input checkerboard are straightened.

B. Using what you have learned from A. take an image of the checkerboard and undistort it. Now using the facilities provided by the openCV calib3d library generate a set of correspondences between the checkboard and the undistorted image. Using these correspondences compute the 3x3 homography (see notes on planar camera model) that maps points on the checkerboard to points on in the image.

C. Finally take a look at the WarpPerspective function in the image processing section of opencv:
http://docs.opencv.org/modules/imgproc/doc/geometric_transformations.html#warpperspective
Using this function, and an image of your choice, project the image such that it appears to lie in the plane of the checkerboard.