Intro to Image Understanding (CSC420)
Assignment 3

Posted: Oct 24, 2015   Submission Deadline : Nov 6, 11.59pm, 2015

Instructions for submission: Please write a document (either pdf, doc, etc) with your solutions (include pictures where needed). Include your code inside the document.

Max points: 12, max extra credit points: 1.5

1. [**2 points**] A robber left his/her shoe behind. Police took a picture of it, see SHOE.JPG. Estimate the width and length (in centimeters) of the shoe from the picture as accurately as possible!

2. [**3 points**] The goal of the exercise is to locate the Halloween toy in TOY.JPG in a collection of eleven test images, 01.JPG, 02.JPG, . . . , 11.JPG. Note that the toy is not a planar object, however, you may assume that its out-of-plane rotation (rotation away from the camera) in the test images is small. Use RANSAC to find the best affine transformation. Visualize the best transformation for the best matching image just like you did for Assignment 2, exercise 2(d). You may use all code you have written for Assignment 2.

3. You are given an image and depth captured with Microsoft Kinect. The file RGBD.MAT contains a variable IM which is the RGB image and DEPTH that contains depth information for each pixel (such an "image" is typically called an RGB-D image). Depth is nothing else but the Z coordinate in camera's coordinate system. To get familiar with it, you can plot it with e.g., IMAGESC(DEPTH) (in Matlab). In this plot, pixels that are red are far away, blue ones are close to the camera, the rest are somewhere in between. Further, you can find a function CAMERA_PARAMS.M which contains the camera's parameters.

   (a) [**1 point**] Compute a 3D coordinate for each pixel (with non-zero depth) in **camera coordinate system**. Plot the computed point cloud (all 3D points). You can use the function PLOT3 (in Matlab). For visually more appealing plots you could also use the function SURF. Include the plot in your solution document.

   (b) [**2 points**] The file RGBD.MAT also contains a variable called LABELS. This variable encodes four objects of interest. For example, IMAGESC(LABELS==1) will visualize the first object of interest, IMAGESC(LABELS==4) the fourth one. Thus, all pixels in LABELS that have value 1 belong to the first object, all pixels that have value 2 belong to the second object, etc. To get the x and

y coordinates of all pixels that belong to the first object, you can do: [Y,X] = FIND(LABELS==1);.

For each object, compute the 3D location for all of its pixels. Now compute the geometric center of each object by simply averaging its computed 3D coordinates. Write code that finds the object (among the labeled four) that is **farthest** from the camera (its distance to camera center is the largest). Write also code that finds the object that is the highest above floor. Here you can assume that the image plane is orthogonal to the floor.

(c) [**1.5 point**] How would you find all (labeled) objects that are on top of another object in RGBD2.MAT? (0.5 points for brainstorming, 1.5 points for implementation)

4. Attached is an image UM_000038.PNG recorded with a camera mounted on a car. The focal length of the camera is 721.5, and the principal point is $(609.6, 172.9)$. We know that the camera was attached to the car at a distance of 1.7 meters above ground.

   (a) [**0.5 points**] Write the internal camera parameter matrix K.

   (b) [**1 point**] Write the equation of the ground plane in camera's coordinate system. You can assume that the camera's image plane is orthogonal to the ground.

   (c) [**1 point**] How would you compute the 3D location of a 2D point (x, y) in the image by assuming that the point lies on the ground? You can assume that the camera's image plane is orthogonal to the ground. No need to write code, math is fine.

   [FOR FUN (not graded)] Attached is code (in Matlab; sorry Python users) that will render a CAD model at a chosen 3D location. The code expects you to click on a trajectory of points in the image, compute 3D locations of these points using your function that implements 4.(c), and will render a CAD model "driving" in 3D on your trajectory. In the function PROJECT_CAD_DEMO in line 42, you'll find the following: [P3D, NG] = YOURFUNCTION(X,Y,??). Replace this function with your function written above that computes 3D locations of the 2D points (x, y) as well as the normal to the ground called NG. Here P3D is a $n \times 3$ matrix of the 3D points and NG is a 3×1 normal vector. Once you have that, the full demo function should run and you should be able to see a video of your rendering.

   We'll include the most innovative renderings on the class webpage.

5. [**Extra credit: 1.5 points**] Make the CAD model drive on the ground that is not orthogonal to the image plane (that car is on a hill). In this task, you'll need to estimate the incline of the ground with respect to the camera. Do this for image 000402.PNG. You are also allowed to manually mark the necessary (relevant) lines in the image. (If you want to do it automatically, you can help yourself with the function FINDLINES.)