Intro to Image Understanding (CSC420)
Assignment 3

Submission Deadline : October 26 (Sunday), 11.59pm, 2014

Max points: 10, max extra credit points: 3

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. The goal of the exercise is to locate the Halloween toy in TOY.JPG in a collection of twelve test images, 01.JPG, 02.JPG, ..., 12.JPG. For all sub-tasks below, please provide code and an explanation. 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. Check Lowe's SIFT paper, in particular Figure 12 and the text that corresponds to it, to see how Lowe matched non-planar objects: `http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf`.

   (a) **[1 point]** Use your function from the previous assignment to compute SIFT features on all twelve test image as well as the reference image (TOY.JPG). Match the reference image to **each test image** and keep **all** matches. Visualize top 10 matches for **each test image** (your solution document has to have 12 pictures).

   (b) **[1 point]** For **each test image** use the top 3 matches from (a) to solve for the transformation between the features in the two images (reference and test). Which transformation did you use?

   (c) **[2 points]** For **each test image** count the number of inliers for the transformation you computed in (b). Please explain how you computed the inliers. In your document, please rank all the images according to the number of inliers. That is, first show the image which has the most inliers, then the image with second highest number of inliers, etc.

   (d) **[1 point]** Take the test image that had the most inliers. Re-compute the transformation using **all** the inlier matches. Visualize the transformation by plotting the rectangle from the reference image transformed to the second image with the computed transformation (just as you did in the Assignment 2, exercise 2 (d)).

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. 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 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. For visually more pleasant 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.

4. [**Extra credit: 3 points**] In this Exercise, the goal is to render synthetic CAD models into real photographs in a realistic way. The purpose of the exercise is to work out a bit of geometry, and to generate really fun videos. We will follow the setup described in the slides 59+ in Lecture 10, `http://www.cs.utoronto.ca/~fidler/slides/CSC420/lecture10.pdf`. To summarize, you are given the camera intrinsic parameters and an image and you know that the image plane is orthogonal to the ground plane. While the authors of the picture state that the distance above ground is 1.7 meters, for the provided CAD data, a distance 1.9 meters works better.

   You are given most of the code that does really simple rendering and trajectory smoothing (feel free to write your own). The code as well as data is in the folder RENDER_CAD, and the main function is PROJECT_CAD_DEMO. Right now it's written in a way such that:
   PROJECT_CAD_DEMO("UM_000038", 64, 0)
   will read in the image UM_000038.PNG and its camera parameters, and will want to render the CAD model called CAR_064_MESH.MAT. I included some other CAD models, e.g. 177 is a formula 1, and there's also a BED_008_MESH.MAT that contains a CAD model of a bed. If you are not happy with the current collection of the models, more are available here: `http://www.cs.toronto.edu/~fidler/projects/CAD.html`, including the CAD visualization code. Even more models can be found on the 3D Warehouse: `https://3dwarehouse.sketchup.com/`. The last parameter of the PROJECT_CAD_DEMO function is whether you want the frames of your video to be stored in a folder. You can always use the FFMPEG (free) software to combine all frames into a video. Include a few videos as well as code in your solution. This time, CDF will accept zip files.

   In the function PROJECT_CAD_DEMO in line 42, you'll find the following: [P3D, NG] = YOUR-FUNCTION(X,Y,??). Your task is to write an actual function 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.

   Particularly inventive solutions might get even more extra credit. This might include flying cars that spin in the air, or solutions that pay attention to occlusion (if your CAD car is behind a real car, you shouldn't render it), solutions that have multiple objects driving around, solutions that also "write" something fun on the road, just to give you a few ideas. The most inventive solutions will be posted on the class webpage (unless you don't want it, which you should indicate in your solution document).