CALIFORNIA INSTITUTE OF TECHNOLOGY

ME/CS 132a, Winter 2011

Problem Set #2 Due: **Feb. 1st, 2011**

Camera calibration

1. [10 points] Consider a pinhole camera with standard linear model plus radial and tangential distortion. Write down the function that maps a point in 3D space to its projection on the image plane. Derive analytically the effect on the image projection of distorting one of the three parameters f, k_1, and u (use a first-order analysis using linearization).

2. [10 points] Use the *Matlab Calibration Toolbox* to calibrate the *Zhang dataset* and report the parameters obtained.

3. [10 points] (This question uses the data obtained in Q2) For each of the parameters f, k_1, and u, plot the error induced in the projection of the point $(0, 0, 1)$ as a function of small changes in the respective parameter. Does this behavior match your prediction in Q1?

4. [20 points] Implement the linear solution of the projection matrix M, calibration matrix K and pose (R and T) as described in class. Calibrate the data in `CalibrationData.mat` using your solution. Compare the results with those obtained by using the *Matlab Calibration Toolbox* (refer to the third calibration example, `demo_script_oulu_3D.m` on how to use the toolbox given 3D world point coordinates and 2D image point coordinates).

   Note: You can load the data in Matlab by running `load CalibrationData.mat`. The dataset consists of the world coordinates of 50 points (`WorldCoordinates`) and their corresponding projections on the image plane (`ImageCoordinates`).

5. [extra credit, undergraduate level] Calibrate the optics of your own camera (e.g., phone camera or computer webcam). Print your own standard checkerboard pattern and create a calibration dataset. Use the *Matlab Calibration Toolbox* to obtain the calibration parameters.

6. [extra credit, graduate level] The MSL rover Curiosity has arrived on Mars safely. However, on Sol 96, all cameras on board stop functioning due to an unexpected and mysterious event. While most of JPL debates whether it was a dust storm, a meteorite, or alien sabotage, you—the most valuable member of the Computer Vision Group—are given the task of fixing things up. After a long night and too much caffeine, you manage to reactivate one pinhole camera, but you discover that the optics have somehow changed, because the images you receive are not coherent with the calibration parameters that you estimated two years ago. Design a calibration procedure suitable for a lonely rover which can only take pictures of an unstructured landscape. (Assume that there are no checkerboards, monoliths, or analogous known structures on Mars, and that the camera is mounted in such a way that it is not possible to take pictures of part of the robot to use as a calibration pattern.)

   Hint: There was no damage to the actuators, and you are given full control of the rover.

Feature detection and matching

This part of the homework asks you to implement and test a feature detection and matching algorithm and analyze the performance of your implementation using the provided test images.

The problems in this sections refer to the following symbols. To make grading easier, please use the same notation in your answers and in your code.

symbol	*units*	*source code variable*	*meaning*
H_{W}	pixels	`H_window`	window size for the calculation of H
f_{thr}	percentage	`f_threshold`	cut off threshold for f
W	pixels	`search_window`	window size for the local maximum search

1. [25 points] Implement a Harris corner feature detector.

   Instructions: Implement the detector according to the guidelines below. Apply it to the Mars Hill dataset by computing 400 feature points for each frame. Generate feature point images for all frame. Choose a set of parameters, and defend your choice noting the tradeoffs involved. Plot the detected feature points in a copy of the original input image and save the feature image for each frame.

   Implementation guidelines: The input to your program is an image file and an integer n representing the number of features to be returned. The output are "feature point images": a copy of the original image with the features highlighted by markers (see the lecture notes for examples).

   Compute the intensity gradient by convolution with a [-2 -1 0 1 2] filter, and weight the SSD scores using a Gaussian kernel. Implement the cut-off threshold f_{thr} for f as a percentage between the minimum and maximum value of f in the image. For example, $f_{\mathsf{thr}} = 50\%$ corresponds to the average of the minimum and maximum value of f.

   Implement the local maximum search as a search over a squared window of size W centered on each feature point candidate. Output the best n points with the highest scores as your detected features. Verify your detector with the Chess test image to make sure it works before trying it on more complex images.

   Reading/writing images: if using Matlab, we recommend using the built-in image functions (imread, imwrite, plot, etc). For other languages, OpenCV is a fine choice.

2. [25 points] Implement a feature matching algorithm.

   Instructions: Implement a feature matching algorithm according to the guidelines below. Calculate matching scores for each pair of consecutive frames using the Mars Hill image sequence (e.g., Frame1/Frame2, Frame2/Frame3). Generate "feature matching images" for each matching pair, using the second frame as the reference frame (e.g., Frame2 for the Frame1/Frame2 matching pair).

   Implementation guidelines: The input to your program is a pair of images, and two lists of features (detected by the previous algorithm). The output are "feature matching images": choose one of the two images that are matched as a reference frame, plot the matching feature points in a copy of the reference image, and mark every matching pair by connecting the two points with a line (see the lectures notes for examples).

   Implement a matching algorithm to match the output of your feature detector for different images. Use a sum of squared differences (SSD) correlator to calculate matching scores and eliminate ambiguous matches by a first to second peak ratio filter. A reasonable value for the ratio distance is 0.6.

3. [extra credit] Analysis of algorithm behavior

   (a) How does W influence the distribution of your features in the image?

   (b) How does H_{W} interacts with f_{thr}? For example, if you keep f_{thr} constant at 1%, what happens if you increase H_{W}?

   (c) What happens if you increase H_{W} dramatically – can you still get 400 features? Are those features 'good' features for image matching?

General guidelines for problem sets involving software

- Send all the software required via email to the TA. It is most helpful if you package code, data, and answers in a .zip or .tgz file.
- We expect your solutions to be in the spirit of "reproducible research"[1]. Include instructions/scripts that allow reproducing your experiments with relatively little effort. For example, include a script "main.m" that calls the other files.
- Your software is evaluated on clarity and elegance as well as correctness. Comment your code. Insert a comment for everything that is not immediately obvious; and do *not* comment what *is* obvious. Think whether you would still understand the code you write after locking it up in a drawer for a year.
- You will be given code examples in a few languages (Matlab, C++, Python), but you are free to use any language with which you are comfortable.
- You are encouraged to use professional libraries (such as OpenCV) for reading/writing files and analogous tasks. However, you cannot use functions which the homework implies you have to write yourself (you cannot use the Harris detector in OpenCV).
- You are not required to follow the "implementation guidelines" perfectly. If you know a better alternative for a certain step, it is ok to use it.
- You are responsible for the parameters you choose. If we give you a "reasonable" value for a parameter that does not appear to work, you should try other values.
- You cannot share code for homework or look at other people's code. You are free to discuss general ideas about the problem (reading aloud your code does not count as discussion).

[1] `http://reproducibleresearch.net/`