

COMP 558 Assignment 2

Prepared by Prof. Michael Langer and T.A.s Chatavut and Florian

Posted: Fri. Oct. 22, 2010

Due: Tues. Nov. 2, 2010 (11:59 PM) – no extensions

The purpose of this assignment is to give you a hands-on experience with high dynamic range images, and with basic image filtering operations that are involved in edge and corner detection.

Questions 1 and 2: High dynamic range images and motion blur

You are given a sequence of images of a single scene shot under various exposure times, along with Matlab starter code for reading the images.

1. **(2 points)** Modify the `readImagesHDR.m` starter code, so that it estimates the image irradiance within each of the three color channels. It should then estimate and print the dynamic range within each of the color channels.
Note: The image irradiance is the exposure divided by the exposure time. The exposure values are specified by the tone map and inverse tone map. See comments in the code.
2. **(5 points)** The program `motionBlur.m` reads in user input for a shutter speed, and an instantaneous pan+tilt rotation of the camera.

- a. Modify this program so that it computes the blurred image that results from a pan-tilt camera rotation during the exposure time.

Hint: Recall the velocity field for rotation from lecture 3. Ignore the second order components of the rotation field, and approximate the velocity field as a constant vector. Compute a 2D convolution of the exposure with an appropriate 2D “motion blur” matrix “b”. Your matrix b should be based on the rotation, the exposure time, and the internal camera parameters.

- b. Choose the shutter speed and camera rotation parameters such that only the motion blurred specularity on the bike helmet is visible in the image, i.e. all other points are dark. What, if any, are the fundamental limits on the length of the blurred specularity in your image? Your answer should be no more than 100 words of text in a **README_Q2.txt** file
- c. In the same **README_Q2.txt** file, briefly describe how you would generalize your solution to (a) so it allowed for second order components of the rotation field, and also allowed for a roll component of the camera? How could you take advantage of the continuity of the rotation fields to speed up your solution? Again, no more than 100 words.

Questions 3 and 4: edge detection

3. (3 points) Fill in the missing code in the program **computeEdgeMap.m** so that edges are found at all orientations. See the requirements in the comments of the code. Your code will be tested using **Q3.m** and **Habitat_67.jpg**.

4. (5 points) To detect edges in Question 3, we used a threshold on the magnitude of the gradient. If that threshold is low, then we will detect many edges that are due to the noise rather than to the signal (signal = true edges). These incorrectly detected edges are called “false alarms”. If we raise the threshold, we can reduce the number of false alarms, but we will also reduce the number of edge “hits”, namely the number of true edges that are detected. (For more on this tradeoff, see http://en.wikipedia.org/wiki/Detection_theory.)
 - a. Modify **edgeConsistent.m** so that it returns a boolean image indicating whether the gradient vector at each pixel is consistent with the gradient vector both of the two neighbors, where the two neighbors are defined in the direction perpendicular to the gradient, i.e. perpendicular to the neighbors defined in Q3. We define the gradient vector at a neighbor of (x,y) to be *consistent* with the gradient vector at (x,y) if the difference in directions of these two gradient vectors is less than $\pi/32$ radians. (Note this implies that the neighbor can have a direction in an interval of $\pi/16$ radians i.e. plus or minus $\pi/32$.)

Use the given **makeTestImage.m** to generate a test image that consists of two squares and a disk. A **groundtruth** edge map is also provided. The script **Q4.m** makes the test image, computes edges using your **computeEdgeMap** from question 3, calls your **edgeConsistent** function, and then plots the percentage of hits and false alarms that are removed by the edgeConsistent function for several thresholds.

- b. Modify **Q4.m** so that it computes values of **hit**, **fa**, **hit_filtered**, **fa_filtered**, which are the number of pixels where there are hits and false alarms in the edge map and in the filtered edge map. Observe (for yourself) how the consistency check affects the hits vs. false alarms.

Instructions:

Submit the required files (**readImagesHDR.m** , **motionBlur.m** , **README_Q2.txt** , **computeEdgeMap.m** , **edgeConsistent.m** , **Q4.m**) along with an optional **README.txt** file that explains any issues you wish to marker to be aware of.

In order to receive full points, your solution code must be properly commented. It is not enough to produce correct answers