

# COMP 558 Assignment 2 : Solution

marked and solutions prepared by Patrick Virie

November 12, 2010

## Question 1 (2/15)

This question relates to the concept of HDR for both scene and camera. To get the dynamic range of a scene using a camera, which is defined by the ratio of max and min irradiance that the scene can produce, we have to invert the pixel intensities back to irradiance using shutter speed and inverse tone map.

$$E = \frac{T^{-1}(I)}{t} \quad (1)$$

However, since the camera has its own dynamic range, in order to approximate the range of the scene's irradiance we need more than one image taken at different shutter speeds. The mask has been employed in each image to ensure that we consider only the pixels that can be trusted (in the range that the inverse tone map works correctly). A pixel might get through the mask more than one time when we look at different images, so people may keep only the latest value or average them. Both are correct, but keep in mind that the dynamic range is very sensitive to small changes (if we double the size of min irradiance, the whole thing will decrease by half), so we might not get the same result.

The only error that I found in this question was from the way of using mask. Some students used the mask to change the intensities of irrelevant pixels to zero and continued to (1). Unfortunately, those zero intensity pixels would produce some irradiances after passing through (1), and had some contributions to the irradiance matrix, which they should not. Also, some students were not sure what they could do with Tonemap.mat and multiplied (1) by ExposureStepSize. This was unnecessary, but no marks were subtracted.

## Question 2 (5/15)

### a. (3/15)

Motion blur happens when more than one pixel responds to an individual ray. More precisely, when the image projection of a scene ray moves during the exposure period, its energy is spread over the pixels on a path. If the image speed is constant, the energy will be uniformly distributed to all pixels in the

path; this can be thought of as an average. Recall from lecture 3, we can compute the first order velocity on the projection plane from:

$$(v_x, v_y)_Y = (f\omega_Y, 0) \quad (2)$$

$$(v_x, v_y)_X = (0, f\omega_X) \quad (3)$$

$$(v_x, v_y) = (f\omega_Y, f\omega_X) \quad (4)$$

To get the distance that a ray will travel in pixel coordinates:

$$(s_x, s_y) = (v_x, v_y)tc = (f\omega_Y, f\omega_X)tc \quad (5)$$

where  $c$  is the unit converter from mm to pixels, and  $t$  is the exposure time.

Since the distance traveled is a constant line and does not depend on the pixel position, the process of averaging exposure can be simulated by convolution where the kernel contains the neighbourhood of involved pixels and is normalized (see the solution code). Keep in mind that the convolution will flip the kernel in both the X and Y axis ( $I(x, y) * K(x, y) = \sum_{a,b} I(x, y)K(a - x, b - y)$ ) and invert the direction of  $vx$  and  $vy$ ; these will nullify the effect of each other and allow us to “draw” a line in the kernel from  $(s_x, s_y)$  directly.

There were several issues that appeared in the submissions for this question. I will start with those I ignored. First, rotating in negative directions: since we do not define the direction of X and Y axis, it is alright to have the inverted result. Second, the motion starting position: the blur line can be drawn either from one edge to another edge or from the center to one edge (diagonal or half diagonal kernel). The difference is only the shifting in the image. Third, some students did not use the provided formula in lecture 3; they reprojected the displacement to the pixel coordinate using the similar triangles. This is not wrong to do. As for the errors for which marks were subtracted: first, there were many people who forgot to convert the displacement to pixel coordinates; second, the indexing error where people swapped X and Y axis (row represents Y, not X); third, some did not normalize the kernel; fourth, the size of  $b$  was fixed, and the line was drawn to this size.

## b. (1/15)

There are two sub questions here. First, to make everything dark except the specularly, you have to increase the shutter speed (reduce the exposure time). However, this adjustment will also shorten the length of blur so you have to increase the angular speeds to compensate.

This question is open-ended. The goal of this question is to force you to think about blurring from motion. Here is an excellent example, quoting from the answer of one student:

“Assuming that we want the blurred specularity to be the only visible object in the image, the length of the blur is constrained primarily by the difference between the irradiance of the specularity and the irradiance of the next brightest area in the image. If the shutter is left open long enough to permit a long blurred specularity, other regions of the image are likely to become non-zero if there are other parts of the scene with irradiance close to that of the specularity. Two extreme cases to consider are when the specularity is the only radiant object in the image, in which case the blur can be extended along an arbitrary path, and when all other points in the scene have an irradiance that is a very small epsilon less than that of the specularity, in which case any extension of the specularity’s blur will permit all other pixels to become non-zero. This basic answer is already over the word limit, and a full answer would have to account for many other factors, including the precise distribution of irradiances in the image and physical properties of the camera’s CCD or the particular means of thresholding exposure values in any computational solution.”

### c. (1/15)

Again, this is one of the good answers:

“If we add the second order components of the rotation fields, the position of the points will play a role in the velocity field: each position has its own computable vector. We can compute this vector for each pixel and associate a blur matrix based on this vector.

To compute the motion blur for a certain pixel, one would need to convolve with the appropriate matrix. If we want to allow for a roll component, we simply need to compute velocities for that transformation and add their contribution to the velocity field associated with each pixel.

Finally, since rotation fields are continuous, we wouldn’t need to compute them all for each pixel. One would simply need to compute the rotation fields for the 4 corner pixels, and the rotation field of a pixel would simply be an interpolation of the rotation fields of the four corner pixels, given the position (closer corners are given a higher weight).”

Note that there are also other good responses, but I chose this answer because it gives explanations to all issues in the question.

### Question 3 (3/15)

This question is straightforward; the requirement is to implement lateral suppression by allowing only the maximum signals in small neighborhood along the gradient direction.

Given an example case, we ask you to implement the rest. There were two common errors that I found. First, the example shows the case of vertical edge, which deals with the boundary condition:  $[-\pi, -\frac{7\pi}{8}] \cup [\frac{7\pi}{8}, \pi]$ . Some students forgot to acknowledge this and copied the ‘or’ operator into the other three cases, which is incorrect. Second, most students did not realize that the pixel

coordinates have the downward Y axis, which means the increasing angle goes in clockwise direction, and it swaps the positions of correct neighbor in the case D1 and D2.

### Question 4 (5/15)

This question consists of two parts: Detection Theory and Edge Consistency. The first part can be done easily once you understand the concept of False Alarm and Hit. Given the groundtruth, we let you fill in the vectors that contain the number of Hit and False Alarms, before and after filtering by edge consistency. There is no particular concern in this part, all of you have done pretty well.

The second part is very similar to question 3, but the implementation is slightly different. Basically, we ask you to look for neighbors in the edge direction (perpendicular to gradient direction) and check their gradients based on the assumption that the gradient should not change abruptly along the edge. The most common error that I found here was the following: the return value of the Matlab function `atan2` ranges from  $[-\pi, \pi]$ , and  $\pi$  should be considered equal to  $-\pi$ . Some students disregarded this fact and compared the angles directly. To make this right is not trivial, the most common way to compare angles is to convert them to unit length vectors in the Euclidean space and check similarity via dot product.