CS 4475 - Computation Photography QUIZ 1

Your Name: ________________________________ Your T-Square ID: ____________________

- ***You must show all work to receive full credit.*** Correct answers with no work shown will receive minimal partial credit, while incorrect answers with correct work shown will receive generous partial credit. Illegible answers are wrong answers.

- Integrity: By taking this quiz, you pledge that this is your work and you have neither given nor received inappropriate help during the taking of this quiz in compliance with the Academic Honor Code of Georgia Tech.

- Academic Misconduct: Academic misconduct will not be tolerated. You are to uphold the honor and integrity bestowed upon you by the Georgia Institute of Technology.
    - Keep your eyes on your own paper.
    - Do your best to prevent anyone else from seeing your work.
    - Do NOT communicate with anyone other than a proctor for ANY reason in ANY language in ANY manner.
    - Do NOT share ANYTHING during the quiz. (This includes no sharing of pencils, paper, erasers or calculators).
    - Do not use notes or books, etc during the quiz.

Problem	Points	Lost	Gained	Running Total	Grader
1	5				
2	10				
3	10				
4	10				
5	5				
Total:	40				

1. (5 points) What is Helmholtz Reciprocity, how does it apply to Dual Photography, and what does it allow us to do? What are the benefits we gain from exploiting it?

   Helmholtz Reciprocity states that a ray of light traveling from source A to destination B will encounter the exact same image transfer function as the reverse ray of light traveling from B to A. This allows us to recover the light transfer function from a single pixel in a projector to a camera pixel, and reverse it, so that we can "capture" an image from the viewpoint of the projector. We gain the ability to reverse the purpose of projectors and cameras, allowing us more flexibility to choose our viewpoints or light source locations, as well as allowing us to use a single projector and capture from multiple viewpoints at the same time with multiple cameras, then reverse it to have "multiple light sources".

   Grading: Helmholtz Reciprocity: +1 for knowing it involves light rays, +2 for demonstrating knowledge of "reversibility" of a ray of light.
   Dual Photography: +1 for knowing that you can swap the projector and camera. (synthesize a photo from the viewpoint of the projector)
   Benefits: +1 for stating that it allows us to use multiple projectors instead of multiple cameras (this is of less benefit speed wise, as projectors are active) +2 for stating that it allows us to use multiple cameras instead of multiple projectors, or +2 for allowing more choices of viewpoints.

2. a: (5 points) If you have two gray scale unsigned 8 bit images, and you subtract them and place the output into a signed 32-bit grayscale image, what math do you use to convert/scale the result so that it will fit into an unsigned 8-bit image again? Show the formulas you would use, and demonstrate how they would be applied using the smallest (-255) and largest (255) input values. You may assume a full scale output range, such that the largest number would be mapped to 255, and the smallest number would be mapped to zero.

   output = (input + 255) / 2

   0 = (-255+255) / 2 = 0 / 2

   255 = (255 + 255) / 2 = 510 / 2

   output = (input - (smallestInput)) * (largestOutput / (largestInput - smallestInput))

   output = (input - (-255)) * (255 / (255 - (-255))

   0 = (-255 - (-255)) * (255 / (255 - (-255)) = 0 * (255 / 510) = 0 * 0.5

   255 = (255 - (-255)) * (255 / 255 - (-255)) = 512 * 0.5

   Grading: formula that adds 255 (+1) and divides by 2 (+2) (or multiples by 0.5)
   +1 for correct zero calculation (from -255)
   +1 for correct 255 calculation (from 255)

b: (5 points) If you have two gray scale unsigned 8 bit images, and you multiply them together and place the output into a signed 32-bit grayscale image, what math do you use to convert/scale the result so that it will fit into an unsigned 8-bit image again? Show the formulas you would use, and demonstrate how they would be applied using the smallest (50) and largest (65000) input values. You may assume a full scale output range, such that the largest number would be mapped to 255, and the smallest number would be mapped to zero.

output = (input - (smallestInput)) * (largestOutput / (largestInput - smallestInput))
output = (input - 50) * (255 / (65000 - 50))
output = input - 50 * 0.0039360969
or
output = input-50 * (1/255) , or (input -50)/ 255 (not exactly right, but worth 1 point)

Grading: -50 from input image (+1)
multiply by (255 / (65000 - 50) +2
(or 1/255 as an approximation) (+1)
50 - 50 = 0 * (255 / (65000 - 50)) = 0 (+1)
65000-50 * (255 / (65000-50)) = 64950 * (255 / 64950) = 255 (+1)

3. (10 points) If you have two 1000x1000 images represented as numpy arrays (A and B), write the python code that would copy a 400x400 region with upper left corner at (100,100) from image B into image A. Use array slicing notation, not iteration through all pixels.

A[100:500, 100:500] = B[100:500, 100:500]
Almost acceptable, but -1 for inefficiencies: A[100:500][100:500] = B[100:500][100:500]

Grading: A = B in correct order (+2)
Start of range is 100,100 (+2)
End of range is 500 (+2) (or start+400)
Colons used properly for slice +2
Commas used to separate dimensions (+2) (+1 for multiple indexing [] [] as above...)
other syntax errors () instead of [], etc... -1

4. (10 points) Given the following 3x3 kernel and single point (impulse) image, show the result of a cross-correlation of the kernel with the image, as well as a convolution of the kernel with the image. (You do not need to reduce fractions.)

5/9	0	0
0	3/9	0
0	0	1/9

Kernel

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	10	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Image

		10 /9				
			30 /9			
				50 /9		

Cross-correlation

		50 /9				
			30 /9			
				10 /9		

Convolution

Grading: +3 for flipping cross-correlation (along both axis).
+2 for correct values (in any diagonal)
+3 for NOT flipping convolution.
+2 for correct values (in any diagonal)

5. (5 points) Explain how finite differences can be used to approximate a derivative. Provide a brief numerical example that includes a kernel that can be applied to an image using cross-correlation.

By subtracting one value from an adjacent value in a finite sample of a continuous function (such as pixels in an image) you can approximate the derivative.

For example: $dF / dx = F(x+1,y) - F(x,y)$

To apply this with a kernel you can use [-1 1] for example. You can also average the finite differences on the left and right side of a pixel with something like [-1 0 1] to get a kernel that has an anchor point in the center (odd number of entries), or even replicate into a 3x3 filter such as a Sobel, etc...

Grading: +2 - Explains that we are using kernel operators to subtract adjacent pixels
+3 - Includes an example kernel that will do so.