# Midterm Examination
# CS 534: Computational Photography

November 3, 2015

**NAME: SOLUTIONS**

| Problem | Score | Max Score |
|---|---|---|
| 1 | | 8 |
| 2 | | 8 |
| 3 | | 9 |
| 4 | | 4 |
| 5 | | 3 |
| 6 | | 4 |
| 7 | | 6 |
| 8 | | 13 |
| 9 | | 7 |
| 10 | | 4 |
| 11 | | 7 |
| 12 | | 10 |
| 13 | | 9 |
| 14 | | 8 |
| Total | | 100 |

1. [8] What are *four* (4) ways that you can control a camera's **exposure** (i.e., intensity or brightness value) in a photograph?

Exposure is the amount of light recorded by a camera. For a given scene point, this depends on the shutter speed, aperture, ISO, and lighting (flash).

2. [8] Fill in each of the following blanks with one of "*smaller*," "*larger*," or "*same*."

(a) [2] The smaller the f-number of a lens, the __smaller____ the **depth of field**.

(b) [2] The shorter the focal length of a lens, the __larger____ the depth of field.

(c) [2] The closer the distance to the object in focus, the ___smaller__ the depth of field.

(d) [2] The faster the shutter speed, the ____same_____ the depth of field.

3. [9] Say I have a **camera** with lens focal length of 40mm, aperture f-number f/5.6, shutter speed 1/500 second, and ISO value 200.

(a) [3] What is the diameter of the lens (in mm)?

f-number = focal-length / aperture-diameter, so
diameter = 40/5.6 = 7.14 mm

(b) [3] What shutter speed should I use to obtain the *same exposure* in a second photo using f/11 instead of f/5.6?

The f/11 aperture is two full stops smaller than f/5.6 (i.e., (3.6)^2/(7.14)^2 = ¼ the area of the f/5.6 aperture), so the shutter speed should be 4 times longer, i.e., 1/125 second.

(c) [3] What shutter speed should I use to obtain the *same exposure* in a second photo using ISO 400 instead of 200?

Doubling ISO, doubles sensitivity and hence doubles exposure, so to obtain the same exposure, make the shutter speed twice as fast, i.e., 1/1000 second.

4. [4] Given a **pinhole camera** with focal length 60, what are the **image coordinates** of the 3D scene point at coordinates (100, 200, 400)?

   u = fx/z = (60)(100)/400 = 15
   v = fy/z = (60)(200)/400 = 30
   So the image coordinates are (15, 30) (assuming image plane is in front of pinhole)

5. [3] Does the **thin lens formula** apply to a pinhole camera for determining which scene points are in focus and which are not? Briefly explain why or why not.

   No, because in a pinhole camera there is only one ray from each scene point that reaches the sensor, meaning that all scene points are in focus, so there is infinite depth of field.

6. [4] If I double the focal length of a camera lens and also move twice as far away from an object I focus on in a scene, what are *two* (2) things that will be different in the two images?

   The fields of view are different (because the sensor size remains the same in both cases) and also the depths of field are different (causing different amounts of blur in the areas in front of or behind the object in focus).

7. [6] What property of the coefficients of a discrete approximation of a **Gaussian filter** ensures that

   (a) [3] regions of uniform intensity are unchanged by smoothing using this filter?

   Coefficients sum to 1 (or, if the coefficients do not sum to 1, then normalization is done by dividing by the sum of the coefficients after convolving the filter with the image)

   (b) [3] the amount of smoothing does *not* depend on orientation of objects in the image?

   Filter is isotropic, i.e., rotationally symmetric.

8. [13] **Laplacian**

(a) [3] Define a 3 x 3 linear filter that can be used as an approximation of the **Laplacian** of an image $f(x, y)$, i.e., $\nabla^2 f = ( \partial^2 f / \partial x^2 ) + (\partial^2 f / \partial y^2 )$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

(b) [3] How can this filter be used to detect **edges** in an image? That is, specify how to create a binary edge image where a pixel's value is 1 if it is at an edge, and 0 otherwise.

Mark a pixel as an edge point (=1) if it is at a zero-crossing, i.e., the sign of the pixel is opposite the sign of one of its 8 nearest neighbors. (Or, if a pixel's value is positive, at least one of its nearest neighbors is less than or equal to zero; similarly, if a pixel's value is negative, at least one of its nearest neighbors is greater than or equal to zero.)

(c) [3] How can this filter be used to "sharpen" an image by **unsharp masking**?

Compute $f - k(f * g)$ where the input, blurred image if $f$, $g$ is the Laplacian filter, * is convolution, and $k$ is a small constant value.

(d) [4] Describe the main steps to compute a **2-level Laplacian pyramid** from an input image. Use a figure to aid your explanation, if desired.

First compute Gaussian pyramid: If original image is f, define the bottom level image as G0 = f. Then blur G0 using a Gaussian filter (e.g., 5 x 5) to obtain G'0. Then subsample G'0 by keeping every other row and every other column to obtain G1, defining the second level of the Gaussian pyramid. Next, blur G1 to obtain G'1, and subsample G'1 to obtain G2, the third level of the Gaussian pyramid. To obtain the 2-level Laplacian pyramid, compute the bottom level by L0 = G0 – G'0. The second level is computed by L1 = G1 – G'1.

9. [7] Say you want to **warp** an image, *I*, into a new one, *J*, by rotating *I* 45° about the origin of the image. This transformation can be described by the mapping from *I*'s (*u*, *v*) coordinates to *J*'s (*x*, *y*) coordinates as: $x = u\cos\theta + v\sin\theta$ and $y = -u\,sin\,\theta + v\cos\theta$

   (a) [4] If pixels in image *I* are all 0s except five 1s at coordinates (0, 0), (1, 1), (2, 2), (3, 3), and (4, 4) (i.e., a diagonal line of five pixels), what is the resulting image *J* after 45° rotation of just the five "1" pixels in *I* using the above transformation and using 0-order (nearest neighbor) interpolation? Use *cos* 45° = *sin* 45° = 0.7.

   Applying the transform we get 1s in *J* at coordinates:
   (0, 0) → (0.7*0 + 0.7*0, -0.7*0 + 0.7*0) = (0, 0)
   (1, 1) → (0.7*1 + 0.7*1, -0.7*1 + 0.7*1) = (1.4, 0) = (1, 0)
   (2, 2) → (0.7*2 + 0.7*2, -0.7*2 + 0.7*2) = (2.8, 0) = (3, 0)
   (3, 3) → (0.7*3 + 0.7*3, -0.7*3 + 0.7*3) = (4.2, 0) = (4, 0)
   (4, 4) → (0.7*4 + 0.7*4, -0.7*4 +0.7*4) = (5.6, 0) = (6, 0)

   (b) [3] What problem(s) does this example demonstrate?

   This example illustrates that digital rotation is not a 1-to-1 transformation, producing a disconnected line after rotation, with "holes" at (2, 0) and (5, 0).

10. [4] Use **bilinear interpolation** to compute the intensity value at point (10.2, 4.5) assuming the four nearest neighbor pixels have the following intensity values: (10, 4) has value 22, (11, 4) has value 42, (10, 5) has value 40, and (11, 5) has value 25.

    First, linearly interpolate between (10,4) and (10,5):
       (22)(.5) + (40)(.5) = 31
    Second, linearly interpolate between (11,4) and (11,5):
       (42)(.5) + (25)(.5) = 33.5
    Third, linearly interpolate between these 2 values:
       (31)(1 - 0.2) + (33.5)(.2) = **31.5**

11. [7] Compute

(a) [4] the **gradient** at the central pixel of the image:

| | | |
|---|---|---|
| 1 | 3 | 10 |
| 2 | 4 | 11 |
| 3 | 5 | 12 |

using the two first derivative (Sobel) filters in the *x* and *y* directions, respectively:

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

The gradient is defined as [∂f/∂x, ∂f/∂y]. Filtering the image with each of the above masks gives these two terms:

∂f/∂x = (1)(-1)+(2)(-2)+(3)(-1)+(10)(1)+(11)(2)+(12)(1) = 36

∂f/∂y = (1)(1)+(3)(2)+(10)(1)+(3)(-1)+(5)(-2)+(12)(-1) = -8

So, gradient is the vector **[36, -8]**.

(b) [3] the **gradient magnitude** at this same pixel.

The gradient magnitude is $\sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} = \sqrt{1360}$ = **36.88**

12. [10] Say you've detected *n* point features in image 1 and *m* feature points in image 2 and you'd like to determine the **2D translation** of image 1 so that it aligns best with image 2.

(a) [3] What is the form of the 3 x 3 homography matrix, **H**, for this special case of 2D translation? Give your answer as a 3 x 3 matrix with letters in positions for unknowns and numbers where a known constant is used.

| | | |
|---|---|---|
| 1 | 0 | *a* |
| 0 | 1 | *b* |
| 0 | 0 | 1 |

where the translation is (*a*, *b*)

(b) [3] What is the minimum number of corresponding points between the two images that are needed to estimate **H** in this case of a 2D translation?

Only one pair of corresponding points is needed because this pair will generate two equations and there are two unknowns (*a* and *b*).

(c) [4] Give two (2) benefits from incorporating the **RANSAC** algorithm as part of the computation of **H**.

RANSAC makes the estimation of **H** more robust to
(i) errors in determining true corresponding feature points, and
(ii) errors in estimating the precise subpixel coordinates of each feature point.
Furthermore,
(iii) using the largest number of consistent pairs (i.e., inliers) to estimate **H** using least-squares will result in the most accurate **H**.

13. [9] **Texture Synthesis**

(a) [6] The **Image Quilting** algorithm for texture synthesis iteratively selects and adds texture blocks to a partially-defined output texture image.

(i) [3] How is a new block selected to add at each iteration?

Use sum-of-squared distance (**SSD) matching** to find the block of pixels in the source image that has smallest total SSD score based on all pixels in the block that overlap with pixels that are already defined in the output image.

(ii) [3] How are seams between adjacent blocks "hidden"?

Seams are avoided by finding the minimum cost path through the region where two adjacent blocks overlap, where the cost at a pixel in the overlap region is equal to the intensity difference between the corresponding pixels. This seam then determines from which block the pixel's intensity value is copied from. Use dynamic programming to find this seam efficiently.

(b) [3] What additional property/term does the **Criminisi best-first filling algorithm** include to improve on the Image Quilting algorithm? Describe qualitatively; no equation(s) required.

Criminisi's method determines the next pixel (and its neighboring block) to fill based on (1) it must be a pixel that is adjacent to some already filled pixels, with the more the number of neighbors already-filled the better; and (2) the already-filled neighbors have an edge between them that extends in the direction of the unfilled pixel, i.e., fill near linear structures first. These two components are defined as a confidence term and a data term, respectively, and then combined (by multiplication) to give a priority score to each unfilled pixel. The pixel with the highest priority score is filled next. The second term uses the gradient of the already filled neighbors.

14. [8] **SIFT Descriptor**

(a) [3] How is the SIFT descriptor made 2D orientation invariant?

The dominant (i.e., most frequently occurring) orientation associated with all the pixels in the neighborhood around a given feature point, computed using the local gradient orientation at each pixel in the neighborhood. Then, all gradient directions are defined relative to this dominant orientation.

(b) [3] Describe how the histogram(s) is (are) are constructed for use in the descriptor.

Histograms of local gradient directions are computed for 4 x 4 blocks of pixels over the 16 x 16 neighborhood centered at each feature point detected. Hence there are 16 blocks arranged in a 4 x 4 array, where each block contains 16 pixels. For each block the gradient orientation is computed relative to the dominant orientation as described in (a). The orientations are quantized into 8 values, using gradient orientation intervals of width 45°, resulting in an orientation histogram with 8 "bins" for each block.

(c) [2] What are the features contained in the descriptor's feature vector?

The 16 orientation histograms computed as described in (b) are concatenated into a feature vector of length 16 x 8 = 128 values.