

# CIS581: Computer Vision and Computational Photography

## Homework: Cameras and Convolution

Due: Sept. 14, 2017 at 3:00 pm

### Instructions

- This is an **individual** assignment. 'Individual' means each student must hand in their **own** answers, and each student must write their **own** code in the homework. It is admissible for students to collaborate in solving problems. To help you actually learn the material, what you write down must be your own work, not copied from any other individual. You **must** also list the names of students (maximum two) you collaborated with.
- You **must** submit your solutions online on [Canvas](#). We recommend that you use L<sup>A</sup>T<sub>E</sub>X, but we will accept scanned solutions as well. Please place your homework (.pdf), code, images and any additional files into the top level of a single folder named <PennKey>.zip
- **Notation Clarification** For notations in all questions below, we denote  $I$  as input image,  $f$  as kernel and  $g$  as output image. In addition, questions are independent of each other except additional notifications.
- **Start early!** If you get stuck, please post your questions on [Piazza](#) or come to office hours!

## 1 Getting Started

### 1.1 Introduction to MATLAB / Python

In this question, you will be required to build upon the [starter code](#) provided, to read in an image and apply the Sobel operator to it. The kernel which highlights the vertical edges when convolved with an image has been given to you in the code. You need to perform the following tasks:

1. Read in the image *Bikesgray.jpg* into the variable *img1*
2. Convolve the image with the given kernel *f1*
3. Display and save the result of the convolution
4. Come up with a kernel *f2* similar to *f1*, but one which causes the horizontal edges to be highlighted
5. Convolve the image *img1* with kernel *f2*
6. Display and save the result of the convolution.

- **Question 1.1:** Please implement the above via MATLAB or Python. Submit your code and the two images generated as a result of the convolution with kernels *f1* and *f2* respectively.

### 1.2 Filtering

Recall the definition of filtering is as follows:

$$g(i, j) = \sum_{m, n} I(i + m, j + n) * f(m, n) = I \odot f \quad (1)$$

where  $I$  is the image,  $f$  is the kernel for filtering.

- **Question 1.2:** Using the  $I$  and  $f$  given below, check if the commutative property ( $I \odot f \equiv f \odot I$ ) holds for the filtering operation. Show **by hand** how you arrived at your answer. Assume zero-padding along the boundary, and 'same' output size.

$$I = \begin{bmatrix} 0.5 & 2.0 & 1.5 \\ 0.5 & 1.0 & 0.0 \\ 2.0 & 0.5 & 1.0 \end{bmatrix} \quad (2)$$

$$f = \begin{bmatrix} 0.5 & 1.0 & 0.0 \\ 0.0 & 1.0 & 0.5 \\ 0.5 & 0.0 & 0.5 \end{bmatrix} \quad (3)$$

**Note:** The matrices for  $I$  and  $f$  given in Eq. 2 and 3 respectively are to be used only for Question 1.2.

## 2 Convolution

Recall the definition of convolution,

$$g = I \otimes f \quad (4)$$

where  $I$  and  $f$  represents the image and kernel respectively.

Typically, when kernel  $f$  is a 1-D vector, we get

$$g(i) = \sum_m I(i-m)f(m) \quad (5)$$

where  $i$  is the index in the row or column dimension.

If the kernel  $f$  is a 2-D kernel, we have

$$g(i, j) = \sum_{m,n} I(i-m, j-n)f(m, n) \quad (6)$$

where  $i$  and  $j$  are the row and column indices respectively.

In this section, you need to perform the convolution **by hand**, get familiar with convolution in both 1-D and 2-D as well as its corresponding properties.

**Note:** All convolution operations in this section follow except additional notifications: 1. Zero-Padding, 2. Same Output Size, 3. An addition or multiplication with 0 will count as one operation.

For this problem, we will use the following  $3 \times 3$  image:

$$I = \begin{bmatrix} 0.0 & 1.0 & -1.0 \\ 2.0 & 1.0 & 0.0 \\ 0.0 & 3.0 & -1.0 \end{bmatrix} \quad (7)$$

You are given two 1-D vectors for convolution:

$$f_x = [-1.0 \quad 0.0 \quad 1.0] \quad (8)$$

$$f_y = [1.0 \quad 1.0 \quad 1.0]^T \quad (9)$$

Let  $g_1 = I \otimes f_x \otimes f_y$ ,  $f_{xy} = f_x \otimes f_y$  and  $g_2 = I \otimes f_{xy}$ .

**Note :**  $f_{xy}$  should be full output size.

- **Question 2.1:** Compute  $g_1$  and  $g_2$  (At least show two steps for each convolution operation and intermediate results), and verify the associative property of convolution
- **Question 2.2:** How many operations are required for computing  $g_1$  and  $g_2$  respectively? Show addition and multiplication times in your result.
- **Question 2.3:** What does convolution do to this image?

### 3 Kernel Estimation

Recall the special case of convolution discussed in class: The Impulse function. Using an impulse function, it is possible to 'shift' (and sometimes also 'scale') an image in a particular direction.

For example, when the following image

$$I = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \quad (10)$$

is convolved with the kernel,

$$f = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (11)$$

it results in the output:

$$g = \begin{bmatrix} e & f & 0 \\ h & i & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (12)$$

Another useful trick to keep in mind is the decomposition of a convolution kernel into scaled impulse kernels. For example, a kernel

$$f = \begin{bmatrix} 0 & 0 & 7 \\ 0 & 0 & 0 \\ 0 & 4 & 0 \end{bmatrix} \quad (13)$$

can be decomposed into

$$f_1 = 7 * \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } f_2 = 4 * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

- **Question 3:** Using the two tricks listed above, estimate the kernel  $f$  **by hand** which when convolved with an image

$$I = \begin{bmatrix} 1 & 5 & 2 \\ 7 & 8 & 6 \\ 3 & 9 & 4 \end{bmatrix} \quad (14)$$

results in the output image

$$g = \begin{bmatrix} 29 & 33 & 10 \\ 62 & 52 & 30 \\ 15 & 45 & 20 \end{bmatrix} \quad (15)$$

*Hint: Look at the relationship between corresponding elements in  $g$  and  $I$ .*

### 4 Edge Moving

Object Recognition is one of the most popular applications in Computer Vision. The goal is to identify the object based on a template or a specific pattern of the object that has been learnt from a training dataset. Suppose we have a standard template for a "barrel" which is a  $3 \times 3$  rectangle block in a  $4 \times 4$  image. We also have an input  $4 \times 4$  query image. Now, your task is to verify if the image in question contains a barrel. After preprocessing and feature extraction, the query image is simplified as  $I_Q$  and the barrel template is  $I_T$ .

$$I_Q = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, I_T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Instinctively, the human eye can automatically detect a potential barrel in the top left corner of the query image but a computer can't do that right away. Basically, if the computer finds that the difference between

query image's features and the template's features are minute, it will prompt with high confidence: 'Aha! I have found a barrel in the image'. However, in our circumstance, if we directly compute the pixel wise distance  $D$  between  $I_Q$  and  $I_T$  where

$$D(I_Q, I_T) = \sum_{i,j} (I_Q(i,j) - I_T(i,j))^2 \quad (16)$$

we get  $D = 10$  which implies that there's a huge difference between the query image and our template. To fix this problem, we can utilize the power of the convolution. Let's define the 'mean shape' image  $I_M$  which is the blurred version of  $I_Q$  and  $I_T$ .

$$I_M = \begin{bmatrix} 0.25 & 0.5 & 0.5 & 0.25 \\ 0.5 & 1 & 1 & 0.5 \\ 0.5 & 1 & 1 & 0.5 \\ 0.25 & 0.5 & 0.5 & 0.25 \end{bmatrix}$$

- **Question 4.1:** Compute two  $3 \times 3$  convolution kernels  $f_1, f_2$  **by hand** such that  $I_Q \otimes f_1 = I_M$  and  $I_T \otimes f_2 = I_M$  where  $\otimes$  denotes the convolution operation. (Assume zero-padding)
- **Question 4.2:** For a convolution kernel  $f = (f_1 + f_2)/2$ , we define  $I'_Q = I_Q \otimes f$  and  $I'_T = I_T \otimes f$ . Compute  $I'_Q, I'_T$  and  $D(I'_Q, I'_T)$  **by hand**. Compare it with  $D(I_Q, I_T)$  and briefly explain what you find.

## 5 Camera Model and Camera Projection

- **Question 5.1 Camera Sensor Size:** Choose a digital camera that you own, for example: a mobile phone. What is the height (mm) and width (mm) of its image sensors (front-facing and back)? You can find this by looking it up on the internet. Please include the web address as a reference. How does the sensor size affect the field of view of the camera?
- **Question 5.2 Pixel Size:** What is the image sensor resolution of the back camera? Compute the size of a pixel in millimeters.
- **Question 5.3 Focal Length:** What is the focal length of your camera (front-facing and back)? You can find this by looking it up on the internet. Please include the web address as a reference. Also, compute the focal length of your back camera by measuring its field of view and using the size of its sensor. How similar are they? *Ignore the difference caused due to autofocus*. How does the focal length of the camera affect its field of view?
- **Question 5.4 Camera Matrix:** Compose the intrinsic camera matrix for your digital camera. Assume the axis skew to be zero.
- **Question 5.5 Depth of an Object:** Using your digital camera, capture a picture of your friend (say  $F$ ) with known height (in meters) standing some distance in front of Benjamin Franklin (say  $B$ ) in front of College Hall at the University of Pennsylvania as shown in the Figure 5.5(a) and Figure 5.5(b). *Please ensure that the camera plane is perpendicular to the ground plane while capturing the image.*



Figure 5.5(a): Front View



Figure 5.5(b): Side View

- **Question 5.5.1:** Given the height  $H_F$  (in meters) of  $F$  and pixel height  $h_F$ , compute the distance from  $F$  to the camera (in meters). *Show all your work.*
- **Question 5.5.2:** Given the height  $H_B$  (213 inches) of  $B$  and pixel height  $h_B$ , compute the distance from  $B$  to the camera (in meters). *Show all your work.*
- **Question 5.6 Dolly Zoom:** Capture two images with different camera positions in the same setting as Question 5.4, taking a few steps,  $\Delta d$  backwards for the second image as shown in Figure 5.6(a).  $F$  and  $B$  should appear smaller in the second image since  $\Delta d$  is positive. Simulate the Dolly Zoom effect by scaling up and cropping the second image as shown in Figure 5.6(b) such that your friend is the same pixel height in both the images.

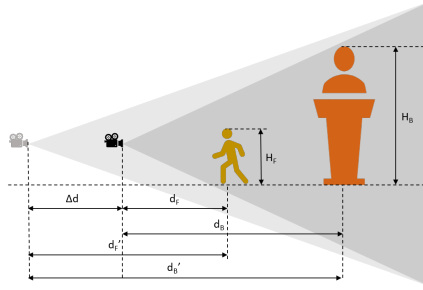


Figure 5.6(a): Geometry of Dolly Zoom

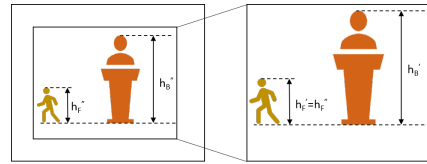


Figure 5.6(b): Scaling up and Cropping

- **Question 5.6.1:** Given pixel height  $h'_B$  in the first image, compute the pixel height of  $B$ ,  $h''_B$  in the second image. *Show all your work.*
- **Question 5.6.2:** Measure the pixel height of  $B$ ,  $h''_B$  in the second image. Are they close to each other?
- **Question 5.6.3:** If we want to increase  $h'_B$  three times while keeping  $h'_F$  the same, what should be the new camera position and focal length of the camera? *Assume that your camera has an optical zoom.*
- **Question 5.6.4: (Optional)** Create a .gif of the Dolly Zoom effect in the given setting by capturing more than two images.