Department of Computer Science
Faculty of Science
Prince of Songkla University

…14.. February 2018

…10:00-11:50…

Lab 5. Arithmetic Operations

LOGIC OPERATIONS AND REGION OF INTEREST PROCESSING

Objectives

- Learn how to perform image addition using the imadd function.
- Explore image subtraction using the imsubtract function.
- Explore image multiplication using the immultiply function.
- Learn how to use the imdivide function for image division.

What You Will Need

1. cameraman2.tif
2. earth1.tif
3. earth2.tif
4. gradient.tif
5. gradient_with_text.tif

Procedure

Four image arithmetic functions of IPT: imadd, imsubtract, immultiply, and imdivide. You could use MATLAB's arithmetic functions (+, −, *, /) to perform image arithmetic, but it would probably require additional coding to ensure that the operations are performed in double precision, as well as setting cutoff values to be sure that the result is within grayscale range. The functions provided by the IPT do this for you automatically.

Image addition can be used to brighten (or darken) an image by adding (subtracting) a constant value to (from) each pixel value. It can also be used to blend two images into one.

1. Use the imadd function to brighten an image by adding a constant (scalar) value to all its pixel values.

```
I = imread('tire.tif');
I2 = imadd(I,75);
figure
subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I2), title('Brighter Image');
```

Question 1 What are the maximum and minimum values of the original and the adjusted image? Explain your results.
Question 2 How many pixels had a value of 255 in the original image and how many have a value of 255 in the resulting image?

2. Use the imadd function to blend two images.

```
Ia = imread('rice.png');
Ib = imread('cameraman.tif');
Ic = imadd(Ia,Ib);
figure
imshow(Ic);
```

Question 3 The image buffers, Ia and Ib, are converted to double precision and added together. How the results are differenced with the image in Ic?

Image subtraction is useful when determining whether two images are the same. By subtracting one image from another, we can highlight the differences between the two.

3. Close all open figures and clear all workspace variables.
4. Load two images and display them.

```
I = imread('cameraman.tif');
J = imread('cameraman2.tif');
figure
```

```
subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(J), title('Altered Image');
```

While it may not be obvious at first how the altered image differs from the original image, we should be able to see where the difference is located after using the imsubtract function.

5. Subtract both images and display the result.

```
diffim = imsubtract(I,J);
figure
subplot(2,2,1), imshow(diffim), title('Subtracted Image');
```

6. Use the zoom tool to zoom into the right area of the difference image about halfway down the image. You will notice that a small region of pixels is faintly white.
7. To zoom back out, double-click anywhere on the image. Now that you know where the difference is located, you can look at the original images to see the change. The difference image above does not quite seem to display all the details of the missing building. This is because when we performed image subtraction, some of the pixels resulted in negative values, but were then set to 0 by the imsubtract function (the function does this on purpose to keep the data within grayscale range). What we really want to do is calculate the absolute value of the difference between two images.

8. Calculate the absolute difference. Make sure Figure 2 is selected before executing this code.

```
diffim2 = imabsdiff(I,J);
subplot(2,2,2), imshow(diffim2), title('Abs Diff Image');
```

9. Use the zoom-in tool to inspect the new difference image. Even though the new image may look the same as the previous one, it represents both positive and negative differences between the two images. To see this difference better, we will scale both difference images for display purposes, so their values occupy the full range of the gray scale.

10. Show scaled versions of both difference images.

```
subplot(2,2,3), imshow(diffim,[]);
title('Subtracted Image Scaled');
subplot(2,2,4), imshow(diffim2,[]);
title('Abs Diff Image Scaled');
```

11. Use the zoom tool to see the differences between all four difference images.

Question 4 How did we scale the image output?
Question 5 What happened when we scaled the difference images?
Question 6 Why does the last image show more detail than the others?

Multiplication is the process of multiplying the values of each pixel of same coordinates in two images. This can be used for a brightening process known as *dynamic scaling*, which results in a more naturally brighter image compared to directly adding a constant to each pixel.

12. Close all open figures and clear all workspace variables.

13. Use immultiply to dynamically scale the moon image.

```
I = imread('moon.tif');
I2 = imadd(I,50);
I3 = immultiply(I,1.2);
figure
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I2), title('Normal Brightening');
subplot(1,3,3), imshow(I3), title('Dynamic Scaling');
```

Question 7 When dynamically scaling the moon image, why did the dark regions around the moon not become brighter as in the normally adjusted image?

Image multiplication can also be used for special effects such as an artificial 3D look. By multiplying a flat image with a gradient, we create the illusion of a 3D textured surface.

14. Close all open figures and clear all workspace variables.

15. Create an artificial 3D planet by using the immultiply function to multiply the earth1 and earth2 images.

```
I = im2double(imread('earth1.tif'));
J = im2double(imread('earth2.tif'));
K = immultiply(I,J);
figure
subplot(1,3,1), imshow(I), title('Planet Image');
subplot(1,3,2), imshow(J), title('Gradient');
subplot(1,3,3), imshow(K,[]), title('3D Planet');
```

Image division can be used as the inverse operation to dynamic scaling. Image division is accomplished with the imdivide function. When using image division for this purpose, we can achieve the same effect using the immultiply function.

16. Close all open figures and clear all workspace variables.
17. Use image division to dynamically darken the moon image.

```
I = imread('moon.tif');
I2 = imdivide(I,2);
figure
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I2), title('Darker Image w/ Division')
```

18. Display the equivalent darker image using image multiplication.

```
I3 = immultiply(I,0.5);
subplot(1,3,3), imshow(I3), ...
title('Darker Image w/ Multiplication');
```

Question 8 Why did the multiplication procedure produce the same result as division?
Question 9 Write a small script that will verify that the images produced from division and multiplication are equivalent.

Another use of the image division process is to extract the background from an image. This is usually done during a preprocessing stage of a larger, more complex operation.

19. Close all open figures and clear all workspace variables.
20. Load the images that will be used for background subtraction.

```
notext = imread('gradient.tif');
text = imread('gradient_with_text.tif');
figure, imshow(text), title('Original Image');
```

This image could represent a document that was scanned under inconsistent lighting conditions. Because of the background, the text in this image cannot be processed directly—we must preprocess the image before we can do anything with the text. If the background were homogeneous, we could use image thresholding to extract the text pixels from the background. Thresholding is a simple process of converting an image to its binary equivalent by defining a threshold to be used as a cutoff value: anything below the threshold will be discarded (set to 0) and anything above it will be kept (set to 1 or 255, depending on the data class we choose).

21. Show how thresholding fails in this case.

```
level = graythresh(text);
BW = im2bw(text,level);
figure, imshow(BW)
```

Although the specifics of the thresholding operation (using built-in functions graythresh and im2bw) are not important at this time, we can see that even though we attempted to segregate the image into dark and light pixels, it produced only part of the text we need (on the upper right portion of the image). If an image of the background with no text on it is available, we can use the imdivide function to extract the letters. To obtain such background image in a real scenario, such as scanning documents, a blank page that would show only the inconsistently lit background could be scanned.

22. Divide the background from the image to get rid of the background.

```
fixed = imdivide(text,notext);
figure
subplot(1,3,1), imshow(text), title('Original Image');
subplot(1,3,2), imshow(notext), title('Background Only');
subplot(1,3,3), imshow(fixed,[]), title('Divided Image')
```

Question 10 Would this technique still work if we were unable to obtain the background image?