**LAPORAN PRAKTIKUM PENGOLAHAN CITRA DIGITAL**

**2. ARITHMETIC OPERATIONS**



**Disusun oleh :**

**Nama : Felix**

**NPM : 2327250059**

**Kelas : IF41**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS ILMU KOMPUTER DAN REKAYASA**

**UNIVERSITAS MULTI DATA PALEMBANG**

**2025**

**TUTORIAL 2. ARITHMETIC OPERATIONS**

**Goal**

The goal of this tutorial is to learn how to perform arithmetic operations on images.

**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.

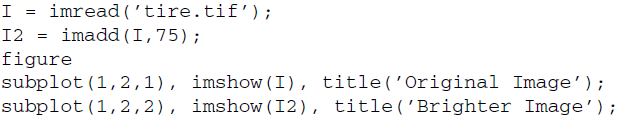
**What You Will Need**

* cameraman2.tif
* earth1.tif
* earth2.tif
* gradient.tif
* gradient\_with\_text.tif

**Procedure**

The IPT offers four functions to aid in image arithmetic: 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.



|  |  |
| --- | --- |
|  |  |

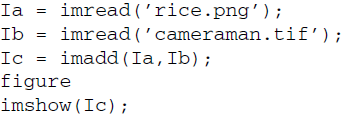
**Question 1** What are the maximum and minimum values of the original and the adjusted image? Explain your results.

|  |  |
| --- | --- |
| Nilai maksimum yang dapat didapatkan adalah 200, nilai minimum sebesar 0, sedangkan gambar yang di adjust memiliki nilai maksimum yang lebih besar yaitu 250 dan minimum 200. |  |

**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?

|  |  |
| --- | --- |
|  | Jumlah piksel pada gambar original saya sebesar 2552202,  Jumlah piksel pada gambar original minimum sebesar 0, |
|  | Jumlah piksel pada gambar adjust maksimum sebesar 6019067, Jumlah piksel pada gambar adjust minimum sebesar 0, |

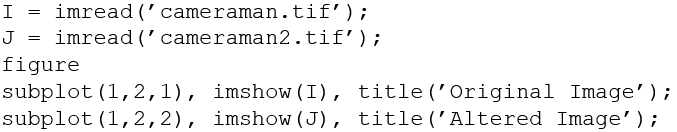
1. Use the imadd function to blend two images.



|  |  |
| --- | --- |
|  |  |

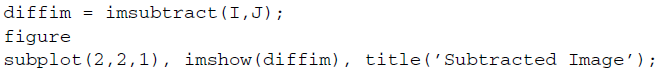
Image subtraction is useful when determining whether two images are the same. By subtracting one image from another, we can highlight the differences between the two.

1. Close all open figures and clear all workspace variables.
2. Load two images and display them.



|  |  |
| --- | --- |
|  |  |

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.

1. Subtract both images and display the result.  
   

|  |  |
| --- | --- |
|  |  |

1. 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.
2. 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.

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

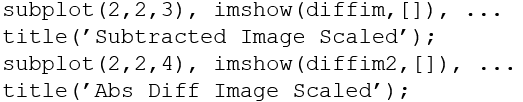


|  |  |
| --- | --- |
|  |  |

1. 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.

1. Show scaled versions of both difference images.



|  |  |
| --- | --- |
|  |  |

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

**Question 3** How did we scale the image output?

|  |  |
| --- | --- |
|  |  |
| Dengan kode *imrezise()* akan mengubah skala gambar atau ukuran gambar, contoh potongan kode. | |

**Question 4** What happened when we scaled the difference images?

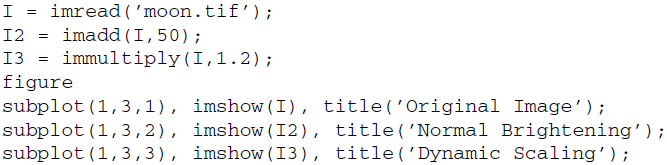
|  |  |
| --- | --- |
|  | Command Error:  Error using imsubtract (line 58)  X and Y must have the same size and class, or Y must be a scalar double. |
| Hal yang akan terjadi jika skala atau ukuran berbeda pada gambar adalah tidak bisa menggunakan function *imsubtract()* dan *imabsdiff()*. | |

**Question 5** Why does the last image show more detail than the others?

|  |  |
| --- | --- |
|  |  |
| Dari penjelasan di atas dan setelah saya tes sendiri saya dapat menyimpulkan bahwa penggunaan *imabsdiff()* lebih detail dikarenakan menggunakan nilai positif sedangkan *imsubtract()* menggunakan nilai negative, dan dari percobaan saya hasil yang dikeluarkan menghasilkan warna berbeda, Ketika menggunakan *imabsdiff* menghasilkan warna yang lebih jelas sedangkan *imsubtract* menghasilkan warna yang hitam putih atau abu-abu. | |

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.

1. Close all open figures and clear all workspace variables.
2. Use immultiply to dynamically scale the moon image.



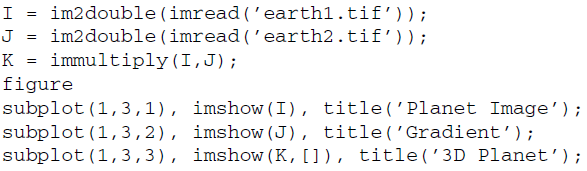
|  |  |
| --- | --- |
|  |  |

**Question 6** When dynamically scaling the moon image, why did the dark regions around the moon not become brighter as in the normally adjusted image?

|  |  |
| --- | --- |
|  | dikarenakan penggunaan immultiply() menaikkan sekala warna dengan cara mengkalikan nilai 1,2, dengan kata lain area gelap dikalikan nilai maka area tetap gelap dan area yang terang maka akan makin terlihat jelas. |

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.

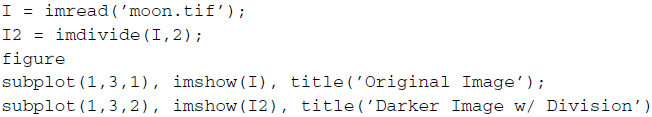
1. Close all open figures and clear all workspace variables.
2. Create an artificial 3D planet by using the immultiply function to multiply the earth1 and earth2 images.



|  |  |
| --- | --- |
|  |  |

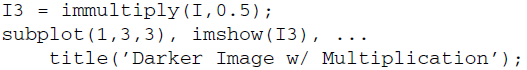
Image division can be used as the inverse operation to dynamic scaling. Image division is accomplished with the imdivide function. When using image division for this purpose, we can achieve the same effect using the immultiply function.

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



|  |  |
| --- | --- |
|  |  |

1. Display the equivalent darker image using image multiplication.



|  |  |
| --- | --- |
|  |  |

**Question 7** Why did the multiplication procedure produce the same result as division?

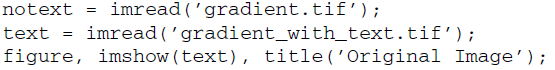
|  |
| --- |
| dikarenakan penggunaan immultiply() dengan cara mengkalikan nilai 0.5, dengan kata lain area gelap dikalikan nilai maka area gelap tetap gelap dan area yang terang maka akan gelap dikarenakan kesluruhan nilai warna di kali 0.5 atau dikali ½ makanya hasilnya sama dengan menggunakan imdivide. |

**Question 8** 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.

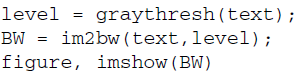
1. Close all open figures and clear all workspace variables.
2. Load the images that will be used for background subtraction.



|  |
| --- |
|  |
|  |

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).

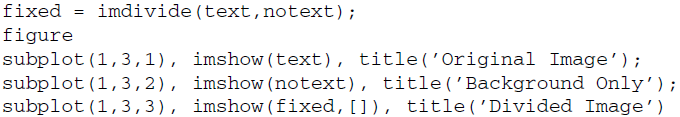
1. Show how thresholding fails in this case.



|  |
| --- |
|  |
|  |

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.

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



|  |
| --- |
|  |
|  |

**Question 9** Would this technique still work if we were unable to obtain the background image?

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
| Menurut saya, teknik ini tidak akan berfungsi dengan baik tanpa gambar latar belakang yang tepat. Ini karena nilai wanra dari notext sebagai pembagi dalam imdivide(). Jika warna pada notext **berbeda-beda** atau tidak sesuai dengan latar belakang asli, hasil imdivide() akan tidak akurat dan detail bisa hilang. Selain itu, perbedaan pencahayaan atau bayangan pada latar belakang akan memengaruhi hasil pembagian. |