Al-Mustaqbal University
College of Sciences
Intelligent Medical System Department

جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الانظمة الطبية الذكية

Lecture (8): Image Algebra

Subject: Image Processing
Level: Third
Lecturer: Asst. Lecturer Qusai AL-Durrah

There are two primary categories of algebraic operations applied to image :

1. Arithmetic operations.
2. Logic operations.

Addition, subtraction, division and multiplications comprise the arithmetic operations, while AND, OR and NOT makeup the logic operations. These operations are done on a pixel –by-pixel basis .

To apply the arithmetic operations to two images, we simply operate on corresponding pixel values. For example to add image I1 and I2 to create I3:

$$\underset{I_1}{\begin{pmatrix} 3 & 4 & 7 \\ 3 & 4 & 5 \\ 2 & 4 & 6 \end{pmatrix}} + \underset{I_2}{\begin{pmatrix} 6 & 6 & 6 \\ 4 & 2 & 6 \\ 3 & 5 & 5 \end{pmatrix}} = \underset{I_3}{\begin{pmatrix} 3+6 & 4+6 & 7+6 \\ 3+4 & 4+2 & 5+6 \\ 2+3 & 4+5 & 6+5 \end{pmatrix}} = \begin{pmatrix} 9 & 10 & 13 \\ 7 & 6 & 11 \\ 5 & 9 & 11 \end{pmatrix}$$

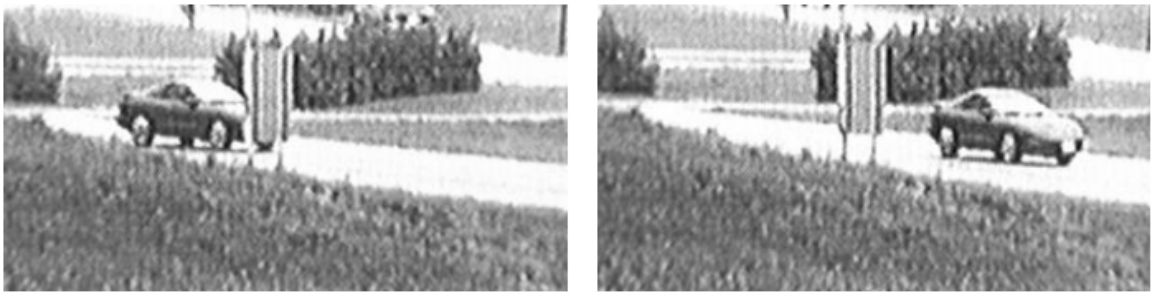
- Addition is used to combine the information in two images. Applications include development of image restoration algorithm for molding additive noise and special effects, such as image morphing in motion pictures.
- Subtraction of two images is often used to detect motion consider the case where nothing has changed in a sense; the image resulting from subtraction of two sequential image is filled with zero (black image). If something has moved in the scene, subtraction produces a nonzero result at the location of movement. Applications include Object tracking.
- Multiplication and Division are used to adjust the brightness of an image. One image typically consists of a constant number greater than one. Multiplication of

the pixel values by a number greater than one will brighten the image (Brightness adjustment is often used as a processing step in image enhancement).

- The logic operations AND, OR and NOT form a complete set, meaning that any other logic operation (XOR, NOR, NAND) can be created by a combination of these basic elements. They operate in a bit-wise fashion on pixel data.

a. First Original image b. Second Original c. Addition of two images

Figure (1): Image Addition



a. Original image b. Same Scene Later

c. Subtraction of scene a from scene b

Figure (2): Image Subtraction.

a. Camera man image

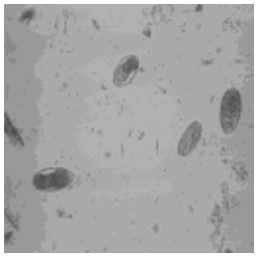
b. X-ray image of hand

c. Multiplication of two images

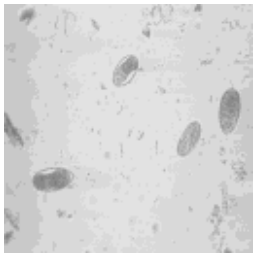
Figure (3): Image Multiplication

Notes:

Other uses of addition include adding a constant offset to all pixels in an image so as to brighten that image. For example, adding a constant value of *50*



yields



Addition is used to combine the information in two images.

Combine

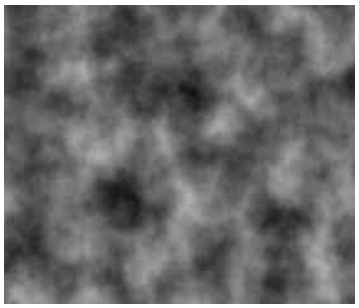
Combines image data from two images (source and destination), letting you specify the areas to be combined, the operations to be performed when combining the data, and which color planes (R with R,G with G, B with B) are used. Combining images based on specified arithmetic operations.

I_1

I_2

$I_3 = I_1 + I_2$

I_1



I_2

$I_3 = I_1 + I_2$

You need to be careful when performing, arithmetic operations that may result in overflows. For example, with an 8-bit unsigned integer (U8) image, adding two pixels with grayscale values of 128 will give a result of 256. The maximum U8 integer is 255, however, so the result will be returned as 255. If you are trying to add several images, the result will be a completely white image. Unless this is the

desired effect, you should store the output as a 16 or 32-bit image to avoid saturation effects.

Subtraction of two image is often used to detect motion.

The pixel subtraction operator takes two images as input and produces as output a third image whose **pixel values** are simply those of the first image minus the corresponding pixel values from the second image. It is also often possible to just use a single image as input and subtract a constant value from all the pixels. Some versions of the operator will just output the absolute difference between pixel values, rather than the straightforward signed output.

How It Works

The subtraction of two images is performed straightforwardly in a single pass. The output pixel values are given by:

$$Q(i,j) = P_1(i,j) - P_2(i,j)$$

Or if the operator computes absolute differences between the two input images then:

$$Q = |P_1(i,j) - P_2(i,j)|$$

Or if it is simply desired to subtract a constant value C from a single image then:

$$Q = P_1(i,j) - C$$

As an example of such change detection



P_1 p_2 $p_3= p_1-p_2$

(*P1)* which shows an image of a collection of screws and bolts. (*p2)* shows a similar scene with one or two differences. If we calculate the absolute difference between the frames as shown in (p_3) then the regions that have changed become clear. The last image here has been contrast-stretched in order to improve clarity.

Multiplication and division are used to adjust the brightness of an image. Multiplication of the pixel values by a number greater than one will brighten the image, and division by factor greater than one will darken the image. It produces a much more natural brightening/darkening effect than simply adding an offset to the pixels, since it preserves the relative contrast of the image better. For instance,

(Im_1) (Im_2) (Im_3)

(Im_1) shows a picture of model robot that was taken under low lighting conditions. Simply scaling every pixel by a factor of 3, we obtain (Im_2) which is much clearer.

However, when using pixel multiplication, we should make sure that the calculated pixel values don't exceed the maximum possible value. If we, for example, scale the above image by a factor of 5 using a 8-bit representation, we obtain (Im_3), all the pixels which, in the original image, have a value greater than 51 exceed the maximum value and are (in this implementation) wrapped around from *255* back to *0*.

The last example shows that it is important to be aware of what will happen if the multiplications result in pixel values outside the range that can be represented by the image format being used. It is also very easy to generate very large numbers with pixel-by-pixel multiplication. If the image processing software supports it, it is often safest to change to an image format with a large range.

before multiply

after multiply

a. Original image

b. Image divided by value<1

c. Image divided by value >1