ARITHMETIC AND LOGIC OPERATIONS

FUNDAMENTALS AND APPLICATIONS

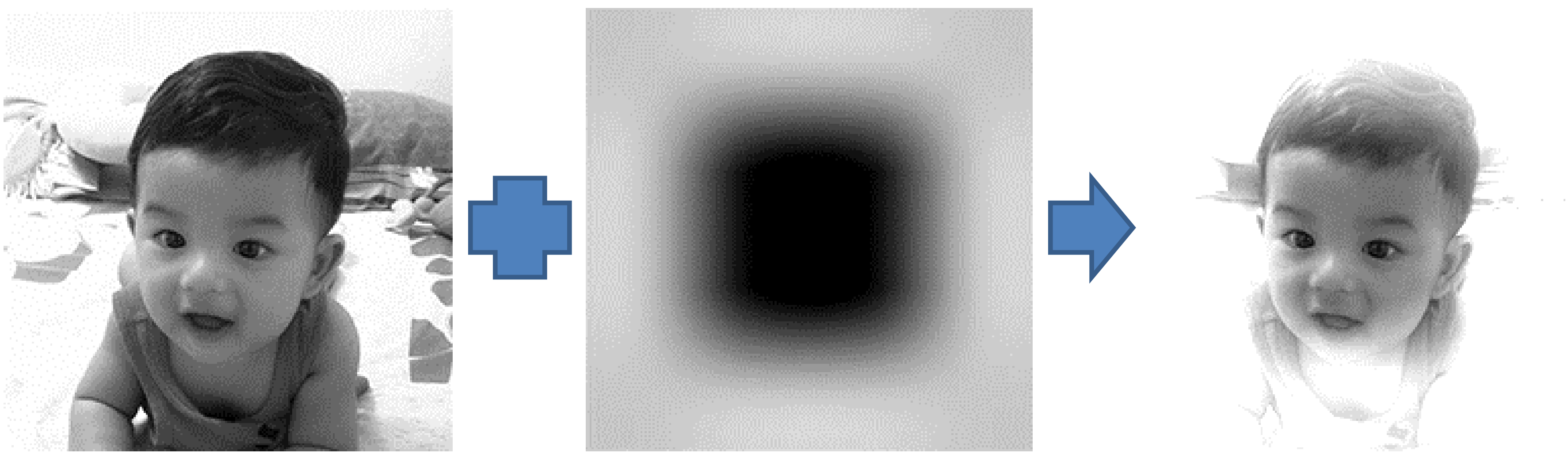
- Arithmetic operations in images perform on a pixel-by-pixel basis.
- Given a 2D array, X, and Y,
- Z obtains by calculating:

$$Z = X \ \Theta \ Y$$

- Where Θ is a binary arithmetic (+, −, ×, /) operator.

Addition Operator

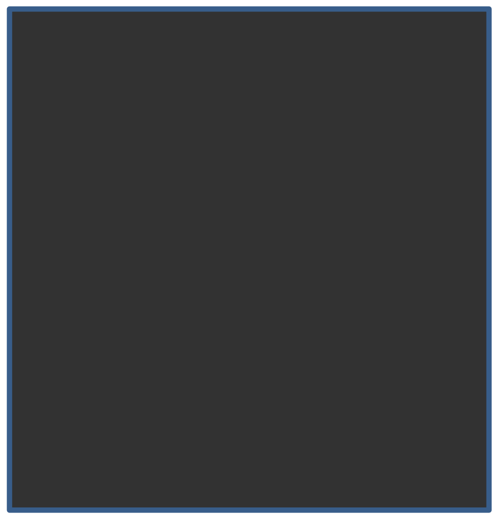
- To blend the pixel contents from two images



X + Y = Z

Addition Operator

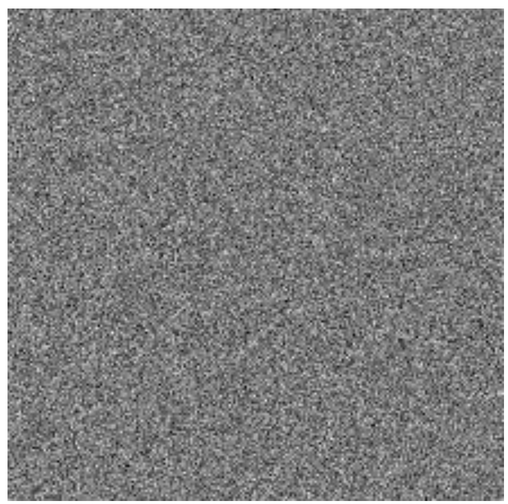
- Or to add a constant value to pixel values of an image.



X + 50 = Z

Noise Addition Operator

- Adding random amounts to each pixel value is a common way to simulate additive noise.



X + noise(0,0.01) = Z

Addition Operator

- Adding two images must be careful with overflow values.
- Two ways to deal with the *overflow* issue:
  - *normalization*

$$g = L_{max}\left(\frac{Z - Z_{min}}{Z_{max} - Z_{min}}\right)$$

  - *truncation*.

Addition Operator

Example:

$$X = \begin{bmatrix} 200 & 100 & 100 \\ 0 & 10 & 50 \\ 50 & 250 & 120 \end{bmatrix} \qquad Y = \begin{bmatrix} 100 & 220 & 230 \\ 45 & 95 & 120 \\ 205 & 100 & 0 \end{bmatrix}$$

W = uint16(X) + uint16(Y);
Za = 255*(W-45)/(350-45);
Zb = X + Y; %imadd(X,Y);

$$W = \begin{bmatrix} 300 & 320 & 330 \\ 45 & 105 & 170 \\ 255 & 350 & 120 \end{bmatrix}$$

$$Z_\mathrm{a} = \begin{bmatrix} 213 & 230 & 238 \\ 0 & 50 & 105 \\ 175 & 255 & 63 \end{bmatrix} \qquad Z_\mathrm{b} = \begin{bmatrix} 255 & 255 & 255 \\ 45 & 105 & 170 \\ 255 & 255 & 120 \end{bmatrix}$$

Subtraction Operator

- Used to detect differences between two images.
- Such differences may be due to several factors
  - Such as artificial addition to or removal of relevant contents from the image (e.g., using an image manipulation program)
  - relative object motion between two frames of a video sequence, and many others.

<u>Subtraction Operator</u>

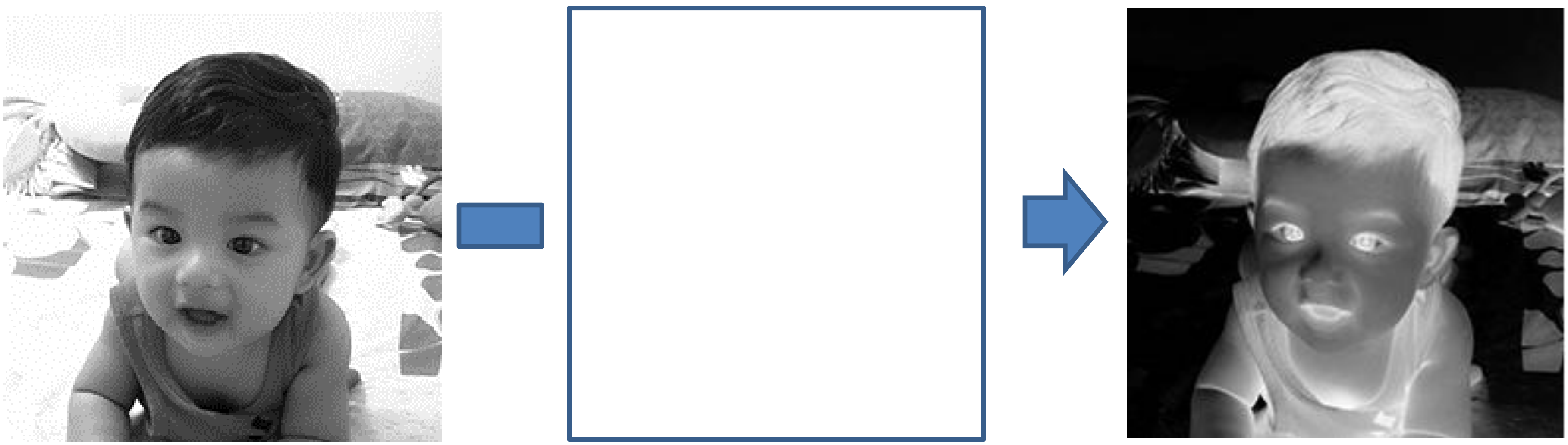
- Subtracting a constant value from an image causes a decrease in its overall brightness, a process sometimes referred to as *subtractive image offset.*

X - 50 = Z

Subtraction Operator

Image subtraction can also be used to obtain the *negative* of an image.

$$Z = L_{max} - X$$



-X + 255 = Z

Subtraction Operator

- Subtracting one image from another or a constant from an image, you must be careful with underflow.
- There are two ways of dealing with this *underflow* issue:
  - **absolute difference** (which will always result in positive values proportional to the difference between the two original images without indicating, however, which pixel was brighter or darker) and
  - **truncating** the result, so that negative intermediate values become zero.

Subtraction Operator

Example:

$$X = \begin{bmatrix} 200 & 100 & 100 \\ 0 & 10 & 50 \\ 50 & 250 & 120 \end{bmatrix} \qquad Y = \begin{bmatrix} 100 & 220 & 230 \\ 45 & 95 & 120 \\ 205 & 100 & 0 \end{bmatrix}$$

Za = X-Y; % imsubtract(X, Y)
Zb = Y-X; % imsubtract(X,Y)
Zc = |X-Y|; %imabsdiff(X,Y)

$$Z_{\mathrm{a}} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 150 & 120 \end{bmatrix}$$

$$Z_{\mathrm{b}} = \begin{bmatrix} 0 & 120 & 130 \\ 45 & 85 & 70 \\ 155 & 0 & 0 \end{bmatrix} \qquad Z_{\mathrm{c}} = \begin{bmatrix} 100 & 120 & 130 \\ 45 & 85 & 70 \\ 155 & 150 & 120 \end{bmatrix}$$

Multiplication and Division Operators

- Multiplication and division by a scalar are often used to perform brightness adjustments on an image.
- *Multiplicative image scaling*—makes each pixel value brighter (or darker) by multiplying its original value by a scalar factor:
  - if the value of the scalar multiplication factor is greater than one, the result is a brighter image;
  - if it is greater than zero and less than one, it results in a darker image.
- Multiplicative image scaling usually produces better subjective results than the additive image offset process described previously.

Multiplication and Division Operators

X

X*0.5

X/0.5

Combining Arithmetic Operations

- To combine several arithmetic operations applied to one or more images may compound the problems of overflow and underflow discussed previously.
- To achieve more accurate results without having to explicitly handle truncations and round-offs, the IPT offers a built-in function to perform a linear combination of two or more images: imlincomb.
- imlincomb computes each element of the output individually, in double-precision floating point.
- If the output is an integer array, imlincomb truncates elements that exceed the range of the integer type and rounds off fractional values.

Combining Arithmetic Operations

Example:

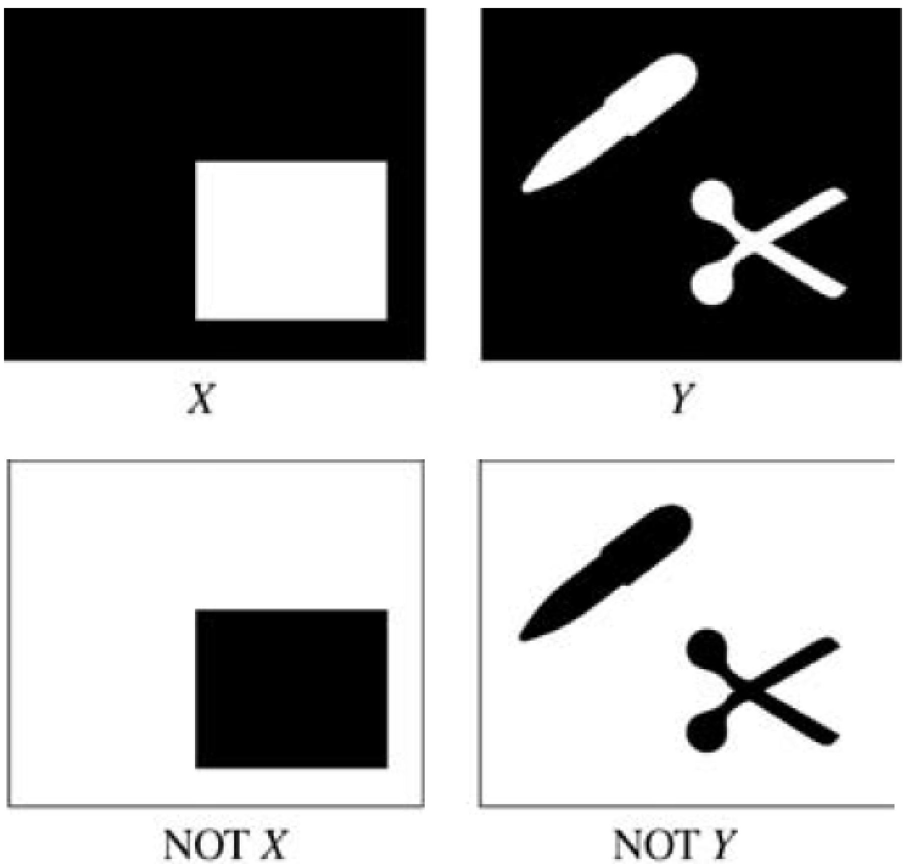
$$X = \begin{bmatrix} 200 & 100 & 100 \\ 0 & 10 & 50 \\ 50 & 250 & 120 \end{bmatrix} \quad Y = \begin{bmatrix} 100 & 220 & 230 \\ 45 & 95 & 120 \\ 205 & 100 & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 200 & 160 & 130 \\ 145 & 195 & 120 \\ 105 & 240 & 150 \end{bmatrix}$$

```
Sa = (X + (Y + Z))/3; % imdivide(imadd(X,imadd(Y,Z)),3)
a = uint16(X) + uint16(Y)
b = a + uint16(Z)
Sb = uint8(b/3)
Sc = imlincomb(1/3,X,1/3,Y,1/3,Z,'uint8')
```

$$S_{\mathrm{a}} = \begin{bmatrix} 85 & 85 & 85 \\ 63 & 85 & 85 \\ 85 & 85 & 85 \end{bmatrix} \quad S_{\mathrm{b}} = \begin{bmatrix} 167 & 160 & 153 \\ 63 & 100 & 97 \\ 120 & 197 & 90 \end{bmatrix} \quad S_{\mathrm{c}} = \begin{bmatrix} 167 & 160 & 153 \\ 63 & 100 & 97 \\ 120 & 197 & 90 \end{bmatrix}$$

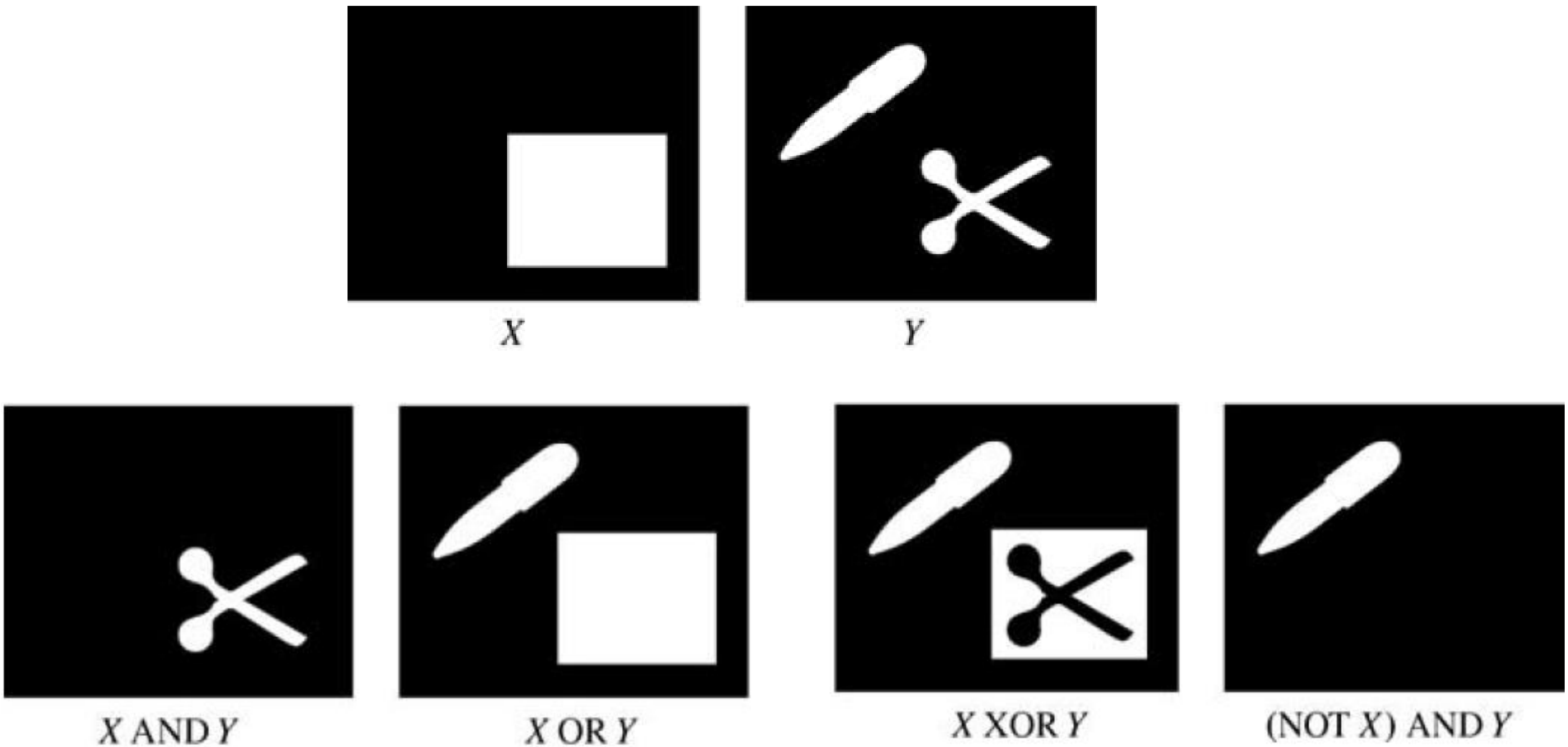
LOGIC OPERATIONS

- They are performed in a bit-wise for each pixel value.
- NOT operator requires only one argument.



LOGIC OPERATIONS

- AND, XOR, and OR operators require two or more operands.

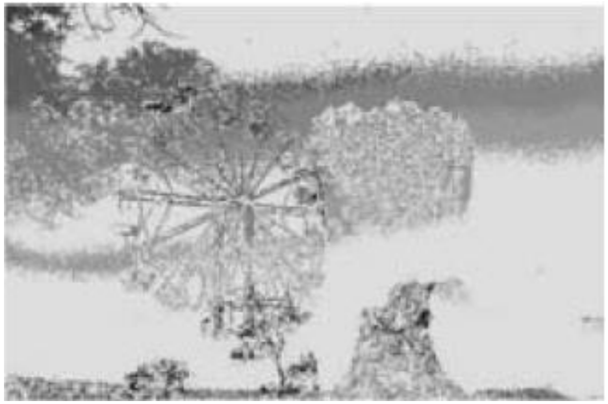


LOGIC OPERATIONS with Grayscale

X

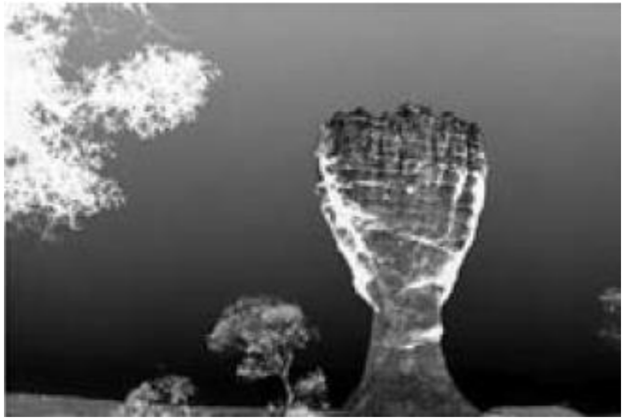
Y

Z = bitand(X,Y)



Z = bitor(X,Y)

Z = bitxor(X,Y)



Z = bitcmp(X,Y)