Transformation

Deepayan Bhowmik

Image Representation

Images are 2D signals.
Signals can be digitised as a structured set of numbers.

- **Image == 2D array or matrix**.
- e.g. image[x][y] = 150;.

```
[[ 8  4  8 34 62 74 73 50 25 13]
 [ 8  3  8 46 82 77 58 34 15 10]
 [ 8  2  9 60 104 82 44 23 12  9]
 [ 9  2 11 74 127 89 34 17 14 11]
 [ 9  1 11 87 148 97 27 12 14 11]
 [ 10  2 13 95 160 101 23  9 14 12]
 [ 10  4 14 88 145 90 19  7 13 11]
 [ 10  8 19 78 123 76 17  8 13 10]
 [ 10 13 27 74 108 67 16  8 12  9]
 [ 10 21 39 72 93 58 16 10 14 10]]
```

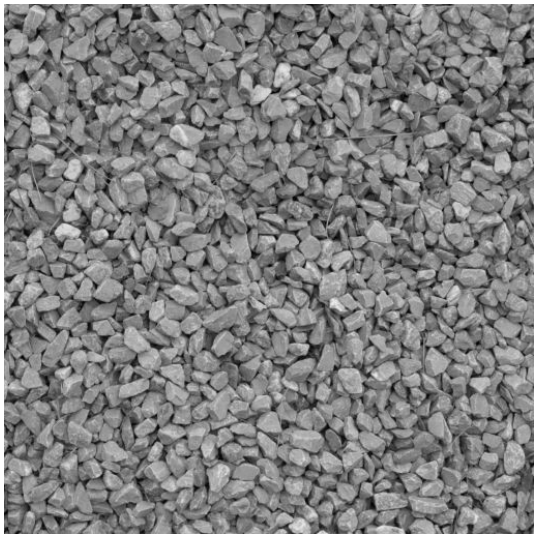
Image Representation

… then can process (or transform) an image by manipulating the corresponding matrix.

A * 0.3

Image Representation

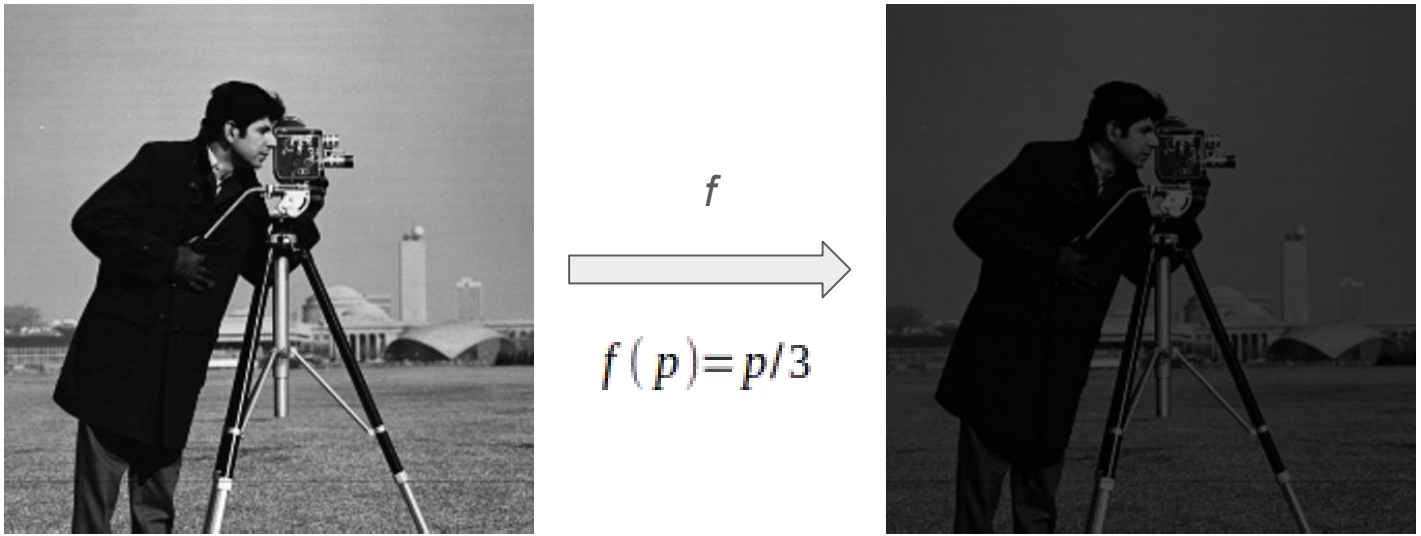
We can process (transform) images by manipulating the corresponding matrices.



A + 0.5*B

Image Representation

… or by processing each pixel, *p*, individually.



Transformation: Types

An image transform may be:

- A point transform:
  - Involving only a single pixel at a time.
- A local transform:
  - Involving the the local image neighbourhood (pixel + those immediately “next to it” - later in course).
- A global transform:
  - Involving the whole image (later in course).

Transformation: Aims

Four categories:

- Remove image degradations introduced during capture.
- Improve image appearance for viewing or further processing (i.e. image enhancement).
- Identify image features for recognition of scene objects (i.e. Image Analysis / Computer Vision).
- Transform image to alternative representation for efficient processing (e.g. Fourier - later in course).

Transformation: Point

Map individual points in the input image to individual points in an output image. Performed as an operation, denoted $\odot$, between two images, I_A and I_B, or between and image and a constant value, C:

$$I_O = I_A \odot I_B$$
$$I_O = I_A \odot C$$

Pixel location *(i,j)* in the output image is computed as follows:

$$I_O(i,j) = I_A(i,j) \odot I_B(i,j)$$
$$I_O(i,j) = I_A(i,j) \odot C$$

Iterative over the image indices for $(i,j) = \{0..w-1, 0..h-1\}$

w = image width, h = image height

Transformation: Addition

Operation: adding a value to each image pixel

- **Contrast Adjustment**: adding a +ve constant value to each pixel increases brightness.

Transformation: Addition

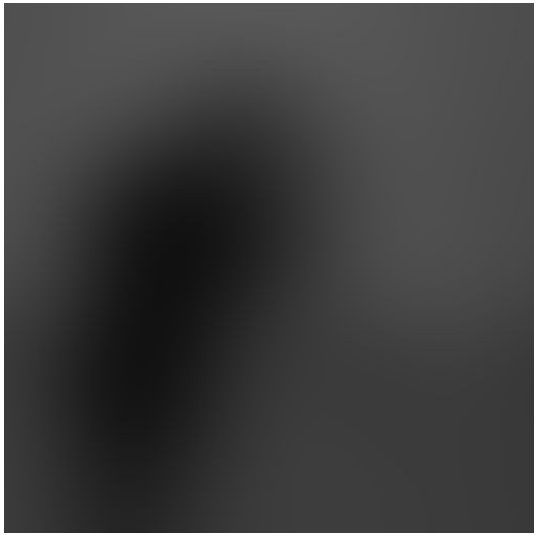
- **Blending**: adding images together produces a composite image of both inputs.



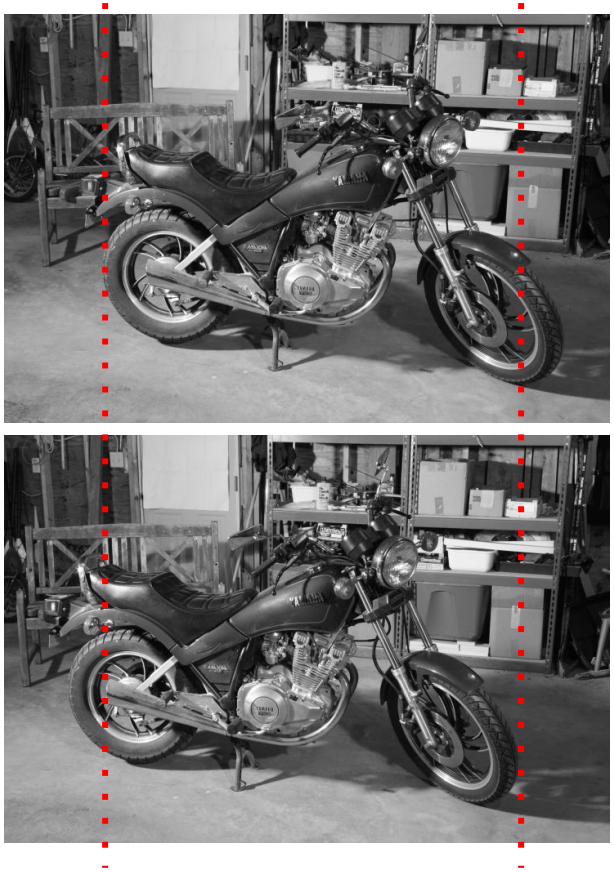
Transformation: Subtraction

Operation: subtracting a value to each image pixel:

- **Contrast adjustment**: as per addition.
- **Image differencing**: subtracting one image from another.



Transformation: Subtraction

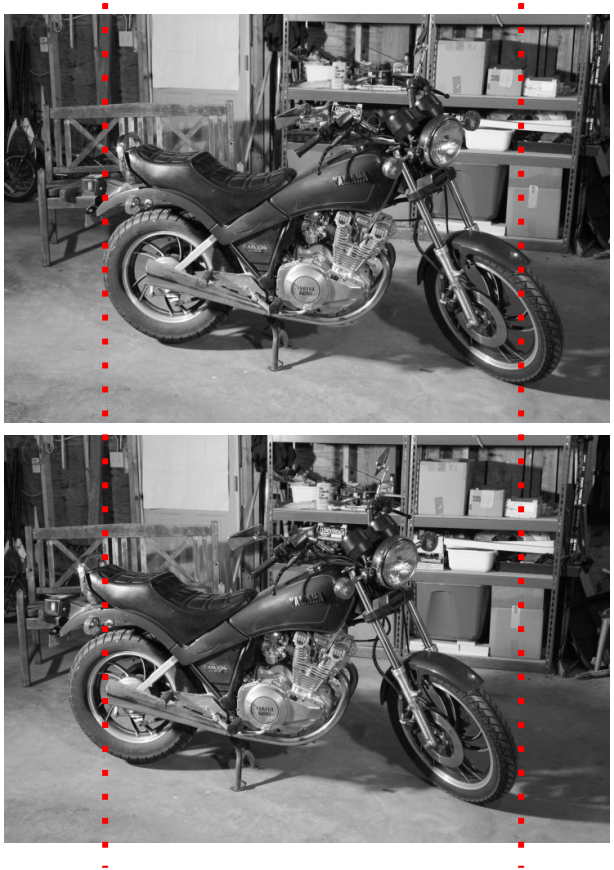


Transformation: Division

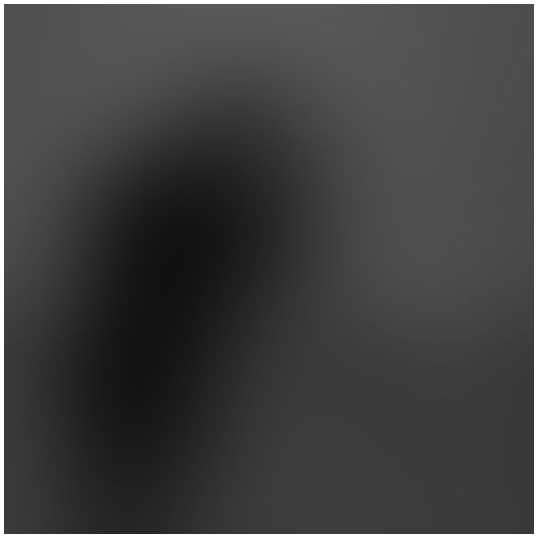
Operation: dividing each pixel value … :

- **Contrast Adjustment**: uniformly scale image contrast:
  - e.g. reduce contrast by 25% = division by 4 (=100/25).
- **Image Differencing**:
  - Dividing image by another: result == 1 where the image pixel values are identical and a value != 1 where differences occur.
    - Image differencing via subtraction more efficient.

Transformation: Division



Transformation: Division



Transformation: Multiplication

Operation: multiplying each pixel value … :

- **Contrast Adjustment**: image colour scaling as per division.

Transformation: Blending

Application of image arithmetic operations

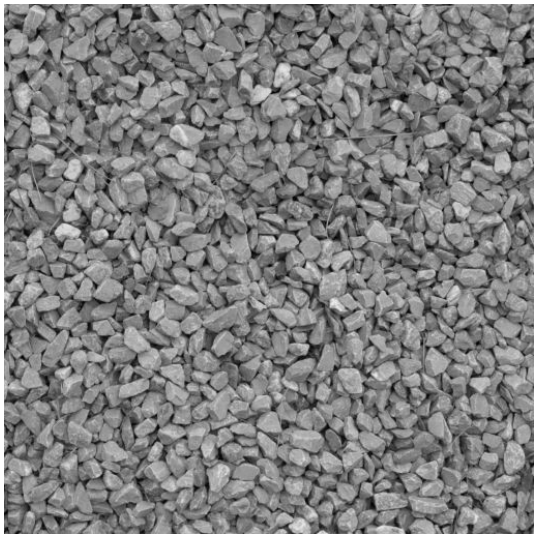
- Produces ghosting or overlay effects between different images.

For N images they can be blended in equal proportions as:

$$I_{output} = \sum_{i} \frac{1}{N} I_i$$

- Alternatively different weights can be used between images to enhance/suppress the features of different images in the final result.

Transformation: Blending



A + 0.5*B

Transformation: Logical NOT

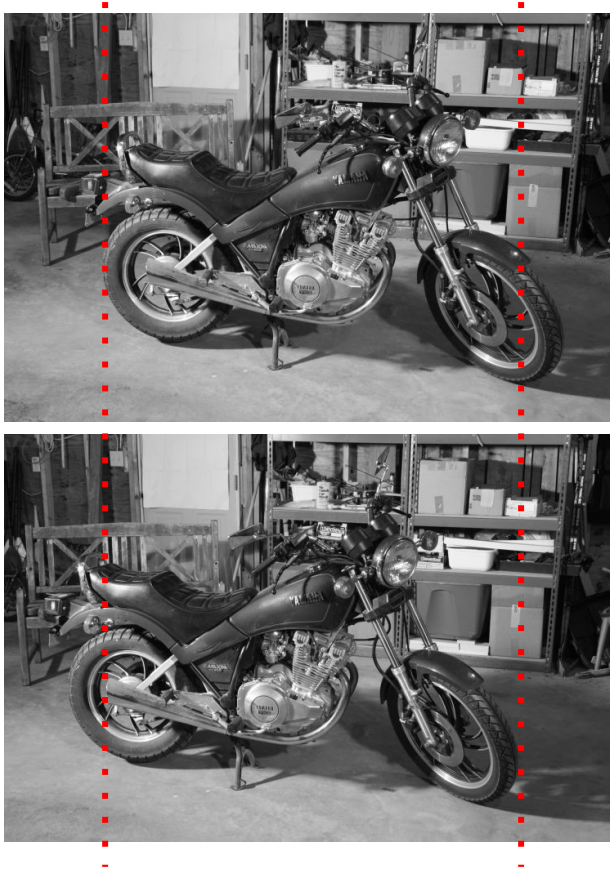
Operation: inverts the image.

- For grayscale (or binary image) dark areas become light and vice versa.
- Colour images differs due to RGB colours – photographic negative effect.

Transformation: Logical AND

Operation: logical AND

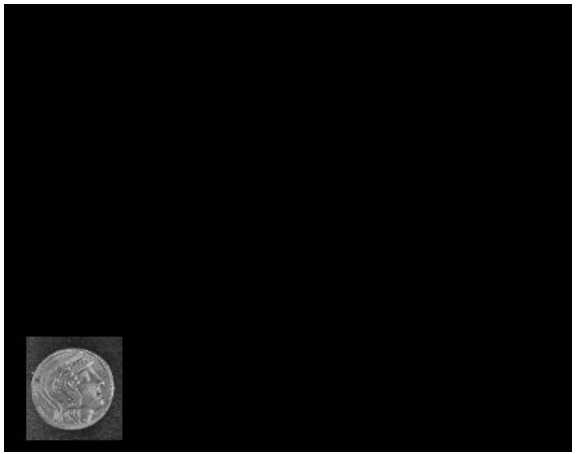
- Used for detecting differences in images.
- Highlighting appropriate regions with a mask.
- Producing bit-planes through an image.



Transformation: Logical AND

Operation: logical AND

- Selecting an individual item or region.



Transformation: Logical AND

Operation: logical AND

- … arithmetic blending.

Transformation: Logical OR

Operation: logical OR

- Useful for processing binary images (0 or 1) images.
- Also to detect common or moved objects.

Transformation: Logical XOR

Operation: logical XOR

- A very useful tool in efficiently detecting image differences.
- Highlights only where changes occur!

Transformation: Logical Operations