Image Enhancement in the Spatial Domain

Some images might be too dark, or too bright, or too noisy. The goal of **Image Enhancement** is to improve these images. These are simple **manipulations** on the image, that will help us improve the contrast and remove the noise. We do this both when we want to make images look better for human consumption and help highlight regions of interest for computer vision.

Image Enhancement is different from image restoration, in that image restoration aims at restoring an image to its original form/fixing degradation rather than making it a better image. Image Restoration works by **modeling** of the degradation and correcting it based on some criteria.

But What Makes a Good Image?

A Good image to a human is a completely subjective thing that is hard to standardize. An image that seems beautiful to one person might not like another. An image that is clear to someone might seem too tiny and ambiguous to another.

However, to a machine, a good image is the one that **gives the best machine recognition results**. Multiple trials and errors might be performed to find the best enhancement approach or technique for an application.

The Spatial Domain

The Spatial Domain is defined as :

> The Domain that contains the pixels of an image.

Spatial domain techniques work directly on the pixels of an image, as opposed to the Frequency domain, in which the operations are done on a signature of the image(eg, the fourier transformation). Spatial domain techniques are more efficient and require less processing resources to implement. Spatial domain processes are expressed as

> g(x,y) = T[f(x,y)]

where f(x,y) is the input image, T is the process or operation on f defined over the **neighborhood** of (x,y), and g(x,y) is the output. The operator can be applied to the whole image or to parts of an image or even sets of images.

Pixel Neighborhood

A Neighborhood in Digital Image Processing is defined as :

> An n*n neighborhood is the area with size n*n around a pixel with the center at (x,y).

Mask/Filter

the process consists of moving the origin of a neighborhood from pixel to pixel and applying T to the pixels in the neighborhood to yield output at that location.

We can determine if we want operations to overlap or not overlap on this region by determining where we move the center during operation. If we are doing certain operations such as inversing or modifying contrast, this could cause certain pixels to be overexposed or the wrong color. Overlapping is not always the best option for an operation.

a special case of mask is point processing

Point Processing/Point-Wise Modification

Point processing is applying a mask to a neighborhood of 1 ie to each pixel in the image . In this case :

> S = T(r)

Where S is the pixels in the output image , r is the pixels in the input image, T is the transform.

Some operations include addition(p+k), Multiplication(pxk), and subtraction(p-k) of pixel values.

An example of point processing is **Thresholding**, where all pixels with gray level above a certain level are turned to white, and all pixels under that level are turned black, Or **Contrast stretching** where all values of r lower than k are compressed(decreased) into a narrow range of s towards black, while all values of r higher than k are stretched(increased) towards white.

Intensity Transformations

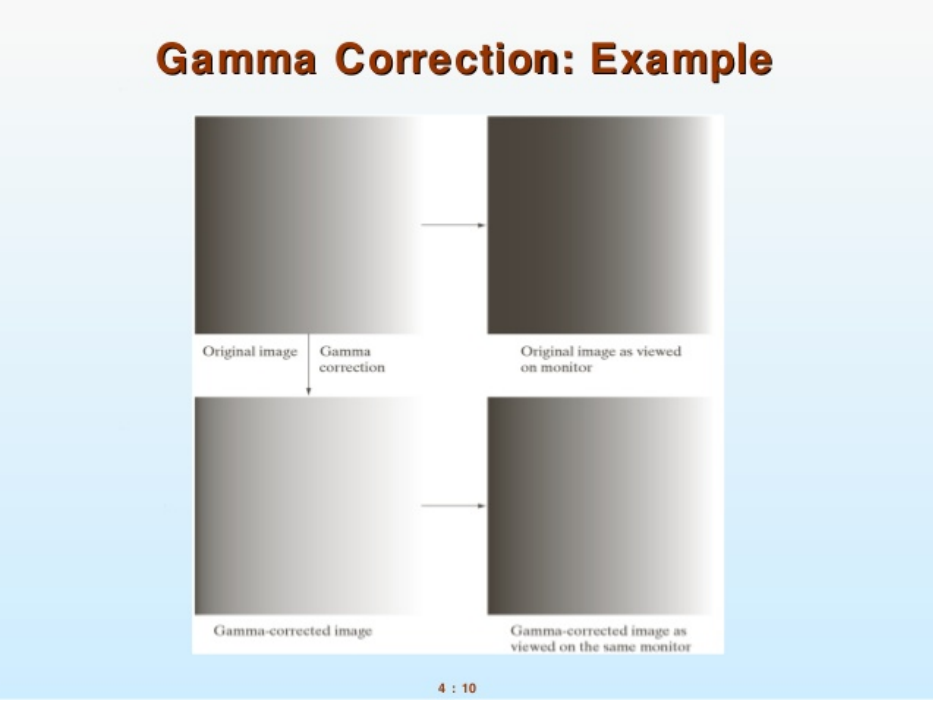
- simplest type of transformation.
- The Values of the pixels are changed individually.
- Basically histogram/point-wise modification/manipulation.
- 1D function to translate input to output.
- The process consists of moving the origin of the neighborhood from pixel to pixel and applying T on the neighborhood 1x1

There are many functions to do this.

Intensity Transformation Functions

- Linear Transformations :
  - Identity Transformation : S = r.
- Negative Transformation : S = L - r, where L is the maximum value in the spatial domain(max intensity in grayscale, for example).
  - Inverse of an inverse is the original image.
  - Useful in the medical field.
- Logarithmic Transformations :
  - General Formula : s = c*log(r+1).
  - c is determined by us. Usually it is 1.
  - We assume r≥1, because we cant calculate log(0).
  - Maps a narrow range of low intensity in the input to a wider range at output.
  - The opposite is true the higher the intensity values of the input
  - Used to expand the value of dark pixels while compressing higher level values.
  - The Opposite is true of inverse log transformation.
  - Compresses the dynamic range of images with large variation in pixel values

    - A classic example of images with pixels with large dynamic range values are Fourier Spectrum images(more than 10^6 possible values). When these values are scaled linearly, the brightest pixels dominate the display, while the lower values are lost. Therefore, we apply the log transform to them, decreasing the range(applying log transform with c=1 on a spectrum with 1.5*10^6 values results in 0-6.2 values).
    - As a results, We cannot see the significant degree of detail as it will be lost in the display.
- Exponential Transformation :
    - General Formula : s = c*r^γ
    - Based on the value of γ, we get different functions mapping the input to the output.
    - Using fractional values maps a narrow range of dark input values to a wider range of output vales.
    - The Opposite is true of higher values.
    - γ = 1 gives us the identity function.
    - Unlike log function, we get a number of transformation curves can be obtained by varying γ.
    - Used for Gamma Correction.
    - Gamma Correction inverts the darkening operation due to some properties in the image
    - Mostly happens in CRT monitors.
    - If we don't do it , the picture will either be too dark or bleached out.
    - Done before inputing the image to the screen.
    - Trying to reproduce colors accurately also requires knowledge of gamma correction since it changes not just the intensity, but also the ratio of R:G:B.




```
- If we decrease gamma too much, the image begins to loose contranst
and is said to be "washed out".
```

- Also Used for general purpose contrast manipulation.
- Piecewise Linear Transformation :
    - More complex(arbitrarily more complex)
    - used for contrast stretching, gray-scale slicing, Bit-Plane slicing.

- Contrast Stretching :
- Simplest piecewise function.
- Improves contrast by stretching intensity values to a span of a desired range of values.
- Automatic contrast Adjustment:
    a. Point operation that modifies pixels such that the available range of values is fully covered.
    b. Algorithm :
        a. Find highest and lowest pixel intensities
        b. Linear stretching of intensity range.

```
- Gray-Level Slicing :
  - Highlights a specific range of gray levels in an image
  - Display a high value of gray level in the range of interest and a
  low value off all other gray levels

- Bit Plane Slicing :
  - Highlights the contribution of specific bits to the
    appearance of the image.
    - Supposed Each pixel is represented by 8 bits
    - lower bits correspond to fine details while higher bits
    correspond to global image details.
    - Higher plane = most significant bit.
    - Useful in image compression.
```