Chapter 2

Fundamentals of Image processing

Point transformation

Look up Table (LUT)

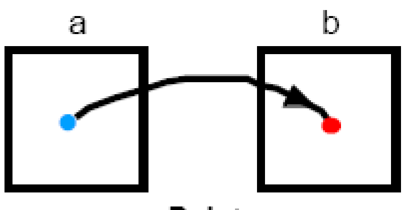
Introduction (1/2)

3 Types of operations in Image Processing

- m: rows index
- n: column index

- **Point to point transformation**

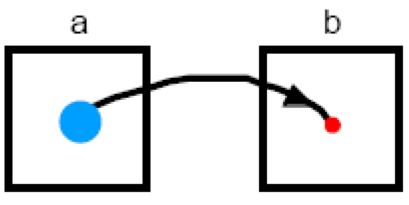
  ● = [$m=m_0$, $n=n_0$]



Point

- **Local to point transformation**

  ● = [$m=m_0$, $n=n_0$]

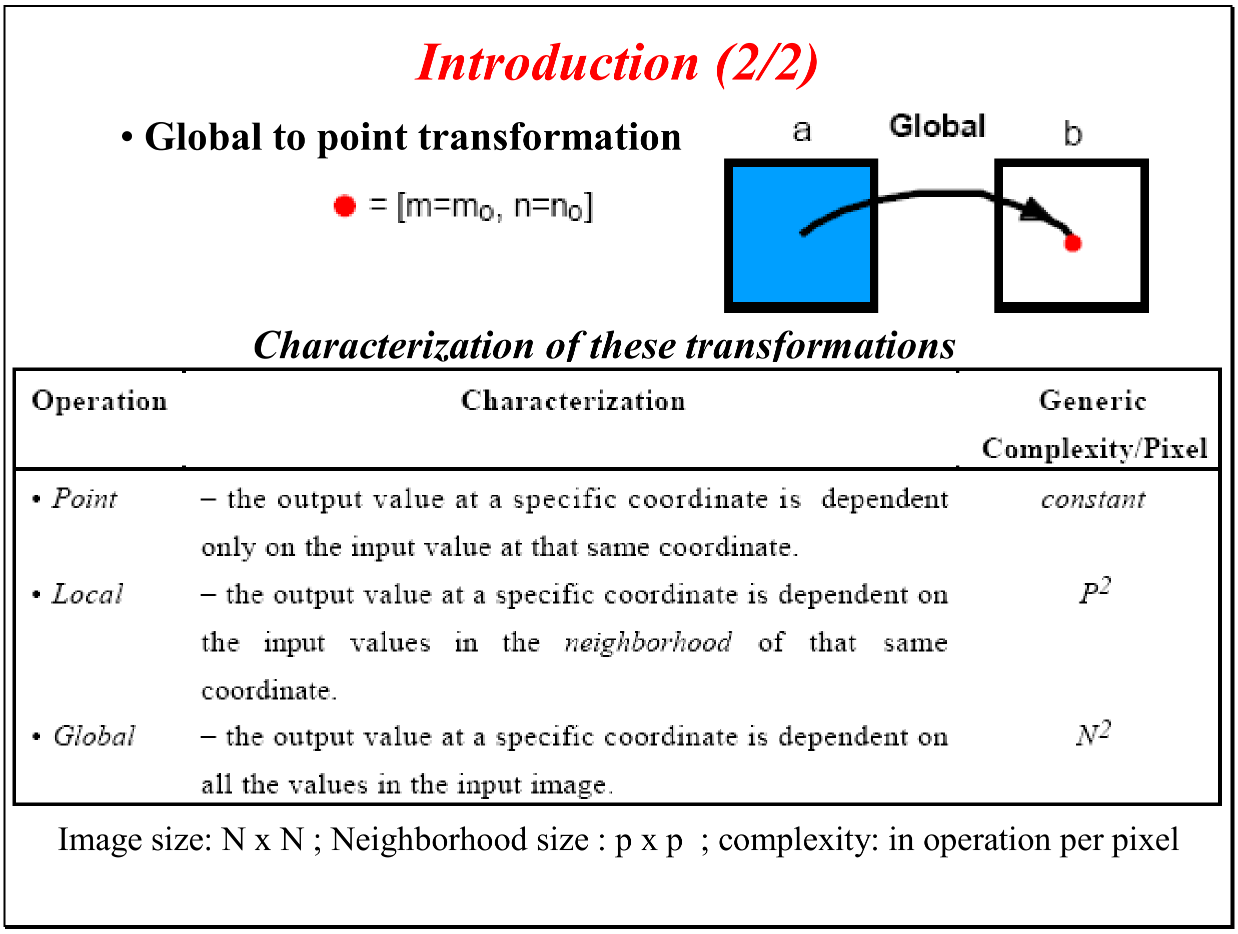


Local

Image Processing is based on three types of operations. The first two of these three types are presented in this figure:

- ***Point to Point transformation*** where the pixel value $P(m_0, n_0)$ of the processed image « ***b*** » is only dependant of the pixel value $P(m_0', n_0')$ of the input image « ***a*** ». Usually, the coordinates (m_0', n_0') are the same at the output and the input: $(m_0', n_0') = (m_0, n_0)$;

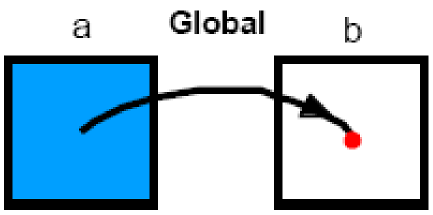
- ***Local to Point transformation***. The value of one element of image ***b*** depends on pixels taken within a window $R(m, n)$ in image ***a***. This window is usually made of a limited number of pixels located around pixel (m_0', n_0') in the input image ***a***: for example a rectangular block of pixels (size $K \times L$), more generally a specific neighborhood of the pixel (m_0', n_0') conserving the same shape and size whatever the coordinates (m_0', n_0') are.



Introduction (2/2)

• **Global to point transformation**

● = $[m=m_0, n=n_0]$



Characterization of these transformations

Operation	Characterization	Generic Complexity/Pixel
• *Point*	– the output value at a specific coordinate is dependent only on the input value at that same coordinate.	*constant*
• *Local*	– the output value at a specific coordinate is dependent on the input values in the *neighborhood* of that same coordinate.	P^2
• *Global*	– the output value at a specific coordinate is dependent on all the values in the input image.	N^2

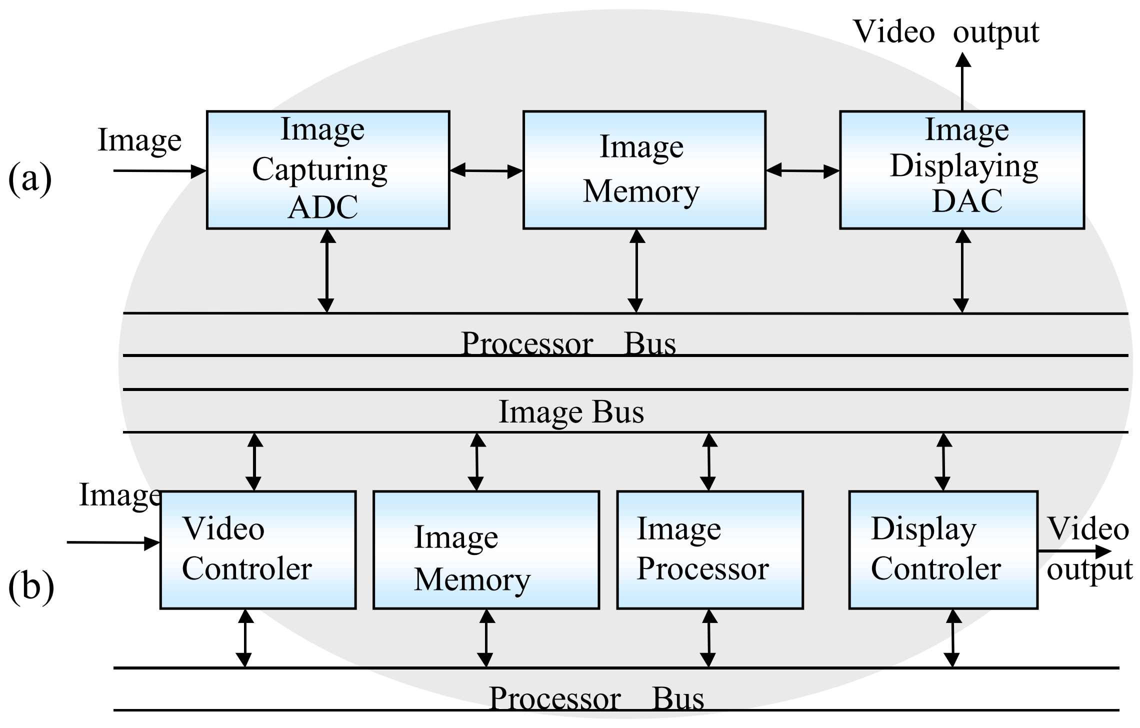
Image size: N x N ; Neighborhood size : p x p ; complexity: in operation per pixel

The last type of transformation is presented in this figure.

- ***Global to Point transformation***. The value of the output at coordinates (m_0, n_0) is dependant of all the pixels in the input image ***a***. This is typically the case when we perform a global transformation of the image into another space. Discrete Fourier Transform (DFT) or Discrete Cosine Transform (DCT) are examples of this type where each frequency component $X(\nu_X, \nu_Y)$ is a function of all the pixels of the image to transform.

If we look at the complexity of the processing in terms of number of operations to perform per pixel at the output, this number is very dependent of the type of processing, from one to N^2 (square image $(N \times N)$), going through P^2 for a window size of $(P \times P)$ for a local operator such as linear filtering by convolution.
Another aspect to look at would be more or less regularity in accessing the image data in the memory of the processor which can slow down the image processing time considerably. It is only mentioned here.

Digital Image Processing System: typical structures

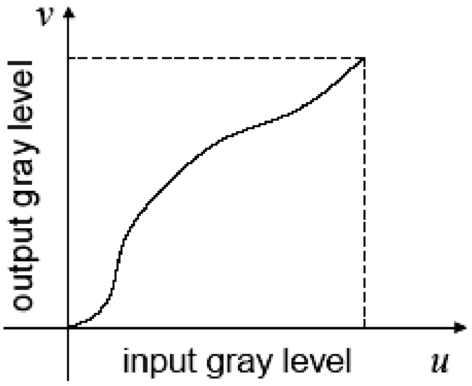


The typical structure of an image processing system is given in this figure. We see at the top a first (simple) version composed of:
the digitalization of analog video signal (sampling & quantization) into digital image data. These data can be stored in an image memory at the sampling rate of the video, transmitted to the second part of the system and/or displayed on a monitor through a DAC;
The image processing is performed by a general purpose processor.
At the bottom, a dedicated image processing unit shows a specific architecture well suited for processing digital images or successive frames of video. It is composed of a dedicated image processor able to perform in real-time operations (linear and non-linear as well) on a limited-size window: (3 × 3) or (5 × 5) window size as well as Point transformation at the input (before processing) and at the output (before displaying).

Point Transformation

Pixel operations

- no memory operation needed
- map a given gray or color level ***u*** of input signal ***i*** to a new level ***v*** of output signal ***o***, i.e. $v = f(u)$
- does not provide any new information
- point transformation can improve visual appearance or make targets easier to detect/extract

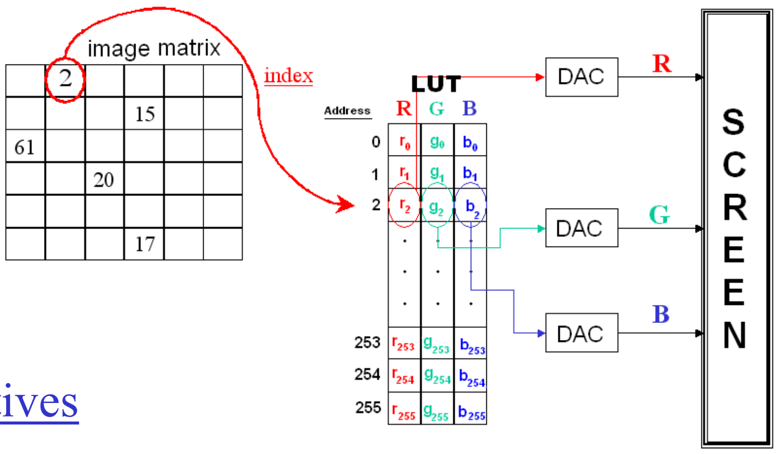


Point transformation can be easily performed with digital values. It corresponds to the transformation of a scalar value into a new one (scalar or vectorial for a color image). It is fully determined by a characteristic function giving the values at the output for each possible value at the input. The main effect of the use of point transformation is to modify the appearance of an image when displaying it immediately after (display of false color image from a grayscale one) or to modify the gray level or color at the input when we want to compensate some non-linearity due to the image sensor. Another interesting use will be in the modification of the image contrast for enhancement purposes.

Point Transformation to display

Principle (for a grayscale image)

Each pixel of input image I_e, of grayscale N_g, is transformed into $T(N_G)$ in the output image I_s through the use of a Look-Up Memory



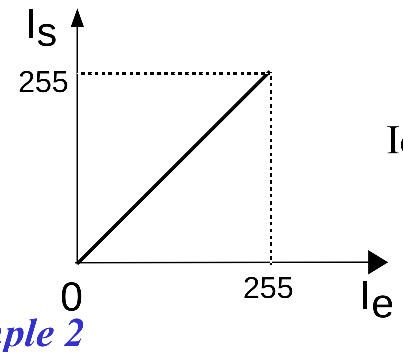
Objectives

- Display range expansion (contrast enhancement)
- Non-linearity correction
- Image binarization
- Segmentation

The basic principle of a point transformation in a (grayscale) digital image is to consider the pixel input value as an address and to read the content of the memory at this address. The content is the transformed value. To do that, original values must have been converted into non negative integers ranging from 0 to typically $2^L - 1$ (quantization and coding of the gray level or color pixel components on 2^L different values each). For a 8-bit pixel coding the size of the memory is of 256 bytes.
A color corresponds to each index “i”. This color is created by the combination $\{r_i, g_i, b_i\}$ of primary colors Red-Green-Blue. The three values $\{r_i, g_i, b_i\}$ are sent to three “digital-to-analog converter (DAC)” to modulate the screen Red-Green-Blue electron guns. By assigning a different image band to each gun, we can create a false color composite image to aid in visual interpretation.
This type of memory is called a Look Up Table (LUT) as its content is computed off-line and not modified during the scanning of all the pixels of the image to transform with the help of the memory. These small-size dedicated memories can be modified during the video synchronization slot time (when the video is not displayed on the TV monitor). This is a good way to realize fading transition between shots in video.
We have usually one LUT at the input and one at the output of a digital image processing system for grayscale images and two sets of 3 LUT’s for color images (one LUT for each of the color components RGB).

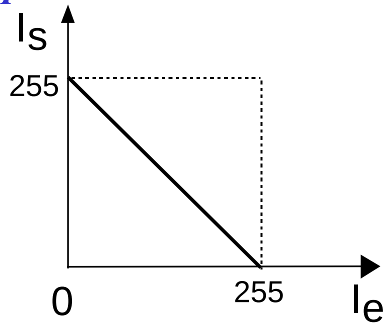
Examples of point transformation

Example 1



Identity transformation : $I_S = I_e$

Example 2



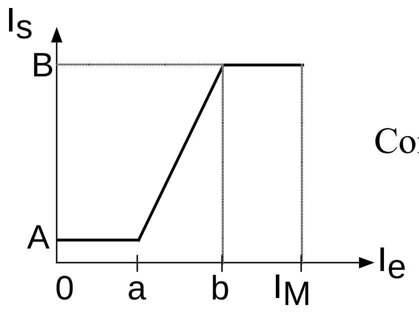
Negative image

⇨ gray-level reverse scaling

If you do not want modify the value, you need to use transparent LUT: the content of the memory at address Ng is Ng. It corresponds to the identity transformation (Example 1). If we want to realize an inverse video for which a low gray-level value is transformed into a high gray-level value and vice versa, the content of the LUT at address Ng is $(2^L$ - Ng –1).

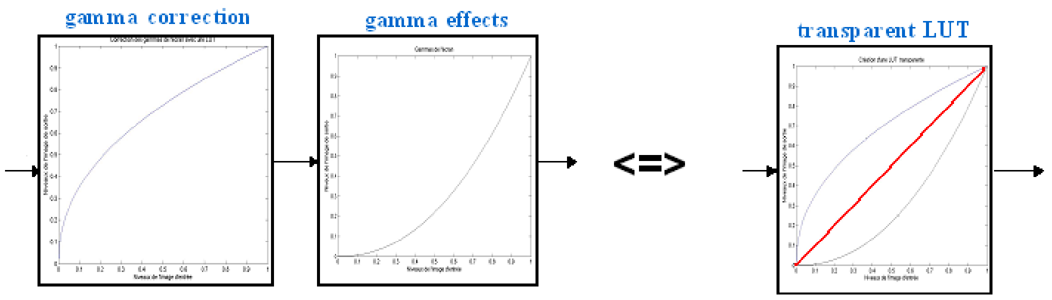
Examples of gray-level transformations

- **Image scaling correction**



Contrast enhancement

- **Sensor correction**



T_2 o T_1 = Identity operator

Two other examples of gray-level point transformations:

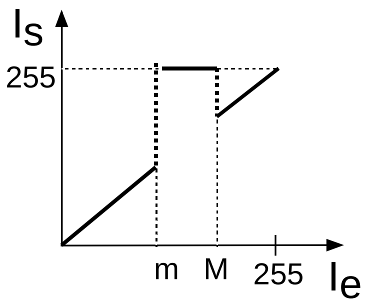
Top figure: all the pixels of the input image I_e with value lower than “a” (respectively higher than “b”), will get the value “A” (respectively “B”) in the output image I_S. The gray-level u_e in image I_e with value included in the range [a, b] will be transformed into a new value v_S between [A, B]. $v_s = \left\lfloor \frac{u_e - a}{b - a}(B - A) \right\rfloor + A$.

The goal is to increase the pixel value range [a, b] up to the range [A, B] and to enhance the contrast. In this case the contrast enhancement factor is worth (B-A) / (b-a).

Bottom figure: shows a typical sensor correction. A cathode ray tube is naturally non linear: light intensity reproduced on screen is a non-linear function of input tension. Gamma correction can be considered as a process which allows us to compensate these effects to obtain a faithful reproduction of the light intensity. Gamma effects are represented by functions $f(x)=x^{\gamma}$, where γ is a real value in the range [2; 2.5] in case of television applications. Note that the concatenation of the two transformations (where T_2 follows T_1) gives an Identity transformation.

Piecewise continuous transformations

Example

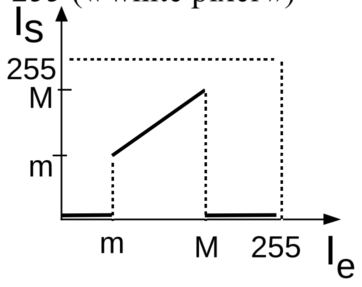


Interest

One performs two thresholdings :

- Values [0, ...,m[and]M,...,255] are kept (the output value is equal to the input value)
- Pixels into the range [m...M] are set to 255 (« white pixel »)

Another transformation :

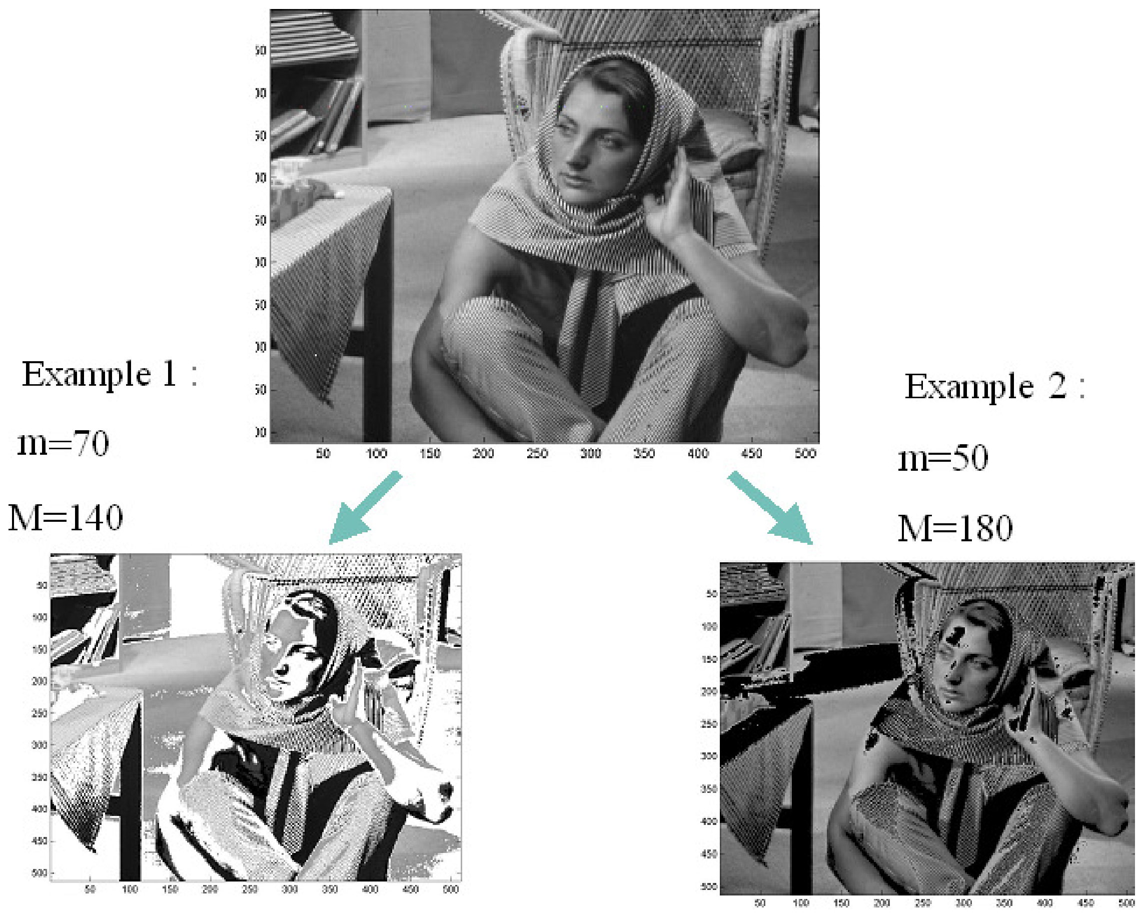


Other examples:

- Top figure: all the pixels of the input image I_e with values in the range [m, M] are set to the value 255 ("white"). For other values (higher than M or lower than m) the input is preserved.

- Bottom figure: the opposite transformation, meaning that only the pixels with values inside the range [m, M] are kept; the others are reset.

These two point transformations are typical examples of piecewise continuous transformations.



The bottom figure shows results for the image *Barbara*. These results are obtained by piecewise continuous transformations.