Image enhancement: point operations and filtering

SCC0251/5830 – Image Processing

Prof. Moacir Ponti

www.icmc.usp.br/~moacir

Instituto de Ciências Matemáticas e de Computação – USP

2020/1

Sumário

Agenda

Image Enhancement

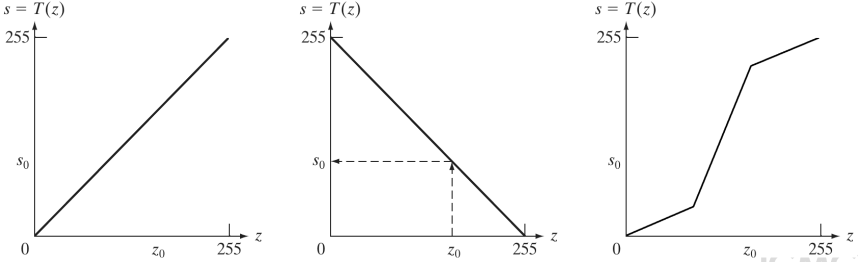
- Modify pixel values for better visualisation;
- Enhancement is often used to obtain images that are better perceived by the human visual system, but can also be useful to serve as input to other algorithms.

Intensity transformation (grey level)

- Altering the grey level intensity of individual pixels;
- Let z be the intensity of an input pixel, and T the transformation:

$$s = T(z),$$

s is the pixel value after transformation.



Identity — Negative/inversion — Contrast modulation

Space domain filtering

- Operations using more than one pixel are often called filtering. In the space domain we have:

$$g(x,y) = T\left[f(x,y))\right],$$

where f is the input image, and g the resulting image. T is an operator defined over the neighborhood of (x,y).

- This way, the transformation can consider either the pixel value (the neighborhood will be 1×1) or also over some arbitrary neighborhood.

Pixel and Neighbours

A pixel p at coordinate (x, y) have four neighbours in horizontal and vertical direction, with coordinates:

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1)$$

This set of pixels is called **4-neighborhood** of p, or $N_4(p)$.

The diagonal neighbours are

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$$

This set is $N_D(p)$.

Pixels $N_4(p)$ with pixels $N_D(p)$ are called the **8-neighborhood** of p, or $N_8(p)$

Agenda

Grey level transformation

- In order to codify this transformation, we design the function T and apply it pixel-by-pixel
- Example:

Inversion (negative)

$T(z) = 255 - z$

Contrast modulation

- Contrast modulation (or adjustment) is an **enhance** method to strech/shrink the range of intensities.
- This linear transformation modifies the range of the input image $[a, b]$ into a new range $[c, d]$:

$$T(z) = (z - a)\left(\frac{d - c}{b - a}\right) + c$$

Logarithmic function

- Shrinks the dynamic range (ratio between the maximum and mininum intensities).

$$T(z) = c \log(1 + |z|)$$

- c is usually defined using the maximum greylevel in the image:

$$c = \frac{255}{\log(1 + R)}$$

- we add 1 to avoid $\log(0)$

Gamma adjustment

- Non-linear operation to enhance pixels of higher intensity.
- γ is the parameter, and it is often used to model the response of display devices (monitors, projectors, etc.)

$$T(z) = cz^{\gamma}$$

- c weighs the result
- γ is often defined between 0.04 and 1.25.

Thresholding

- Can be seen as a segmentation method, but also as a point operation to obtain a mask from an input image.

$$T(z) = \begin{cases} 1, & \text{if } z > L \\ 0, & \text{otherwise} \end{cases}$$

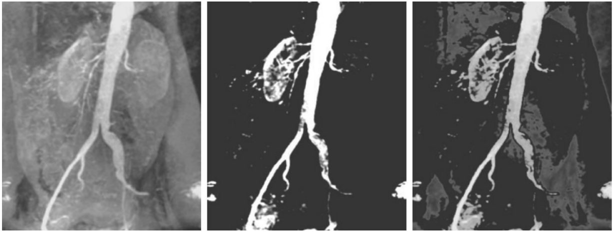
- L is chosen so that it separates only the regions of interest.

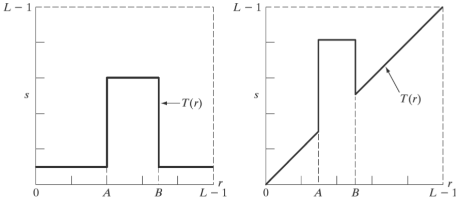
Slicing the grey levels

- Different ranges of intensities may be more relevant in specific contexts. For example:
    - Satellite images: detecting water masses
    - X-rays: enhancing faulty regions in circuits
    - Angiograms: enhancing only vessels and circulatory organs
- The transformation can enhance a range of intensities or selecting bits.

Slicing the grey levels

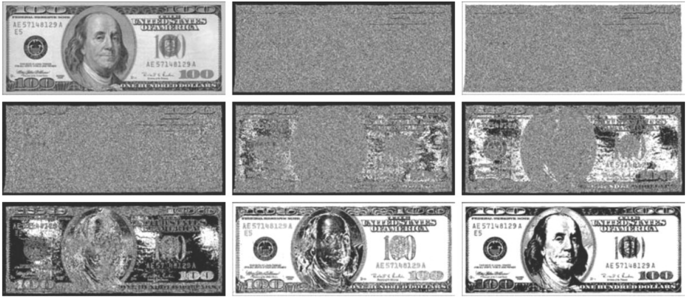
Enhancing interval of intensities





Slicing the grey levels

Bitwise slicing



Histogram

- Information of frequency of each intensity in the image
- Can be seen as
  1. a function $h(k)$, where $k \in [0, L-1]$, and L is the number of possible intensities (or colors) in the image
  2. a vector of size L.
- Often visualised using a bar plot

Example:

0	1	1	1	0
0	1	2	2	0
1	1	1	2	2
1	0	0	0	3
3	3	1	1	1

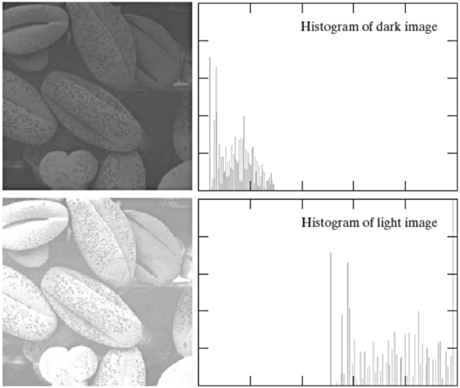
Histogram, Cumulative Histogram and Normalisation

- **Normalised histogram**: each bin of the histogram is divided by the total of pixels, so that the sum is unitary;
- **Cumulative histogram**, $ha(k)$, for each bin k, shows the frequency of all intensities equal or lower than k (shows how much of the total was achieved up to some intensity),
- **Normalised cumulative histogram**: each bin of $ha(k)$ show the percentage of intensities present in the image up to k.

```
0   1   1   1   0
0   1   2   2   0
1   1   1   2   2
1   0   0   0   3
3   3   1   1   1
```

Histogram

- Allow to grasp how the intensities are distributed (globally) over the image



Histogram equalisation

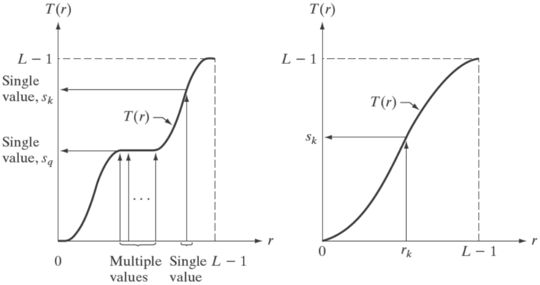
- Produces a non-linear mapping between the input and output pixels
- Uses a transfer function using the image histogram as basis

$$D_B = f(D_A)$$

- D_A is the intensity distribution of the source image
- $D_B = f(D_A)$ is the intensity distribution of the output image

Histogram Equalisation

- We assume the transfer function is monotonic
- The idea is to produce an output that approaches the uniform distribution
- Observe that multiple values in a given input image can be mapped into a single value in the output image, which can be a problem.



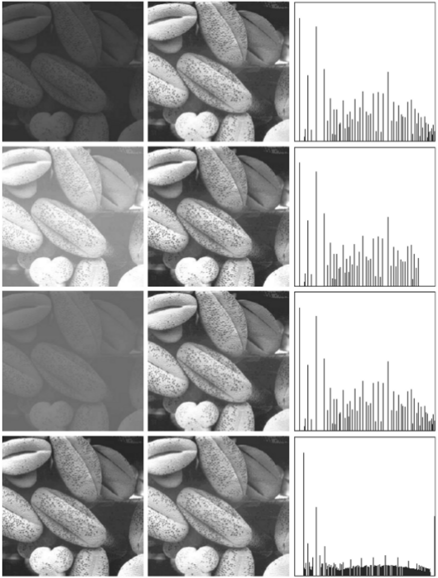
Histogram Equalisation

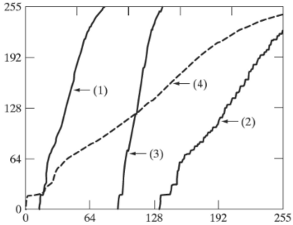
- A simple way to obtain the transfer function is to use the cumulative histogram,
- Using $ha(z)$ we normalise the input pixel z according to the image resolution and quantisation values.

$$s = T(z) = \frac{(L-1)}{MN} ha(z),$$

- $M \times N$ is the image resolution
- $ha(z)$ is the cumulative histogram value relative to the value z
- L is the number of intensities after image quantisation (e.g. 256 for 8 bits)

Histogram Equalisation



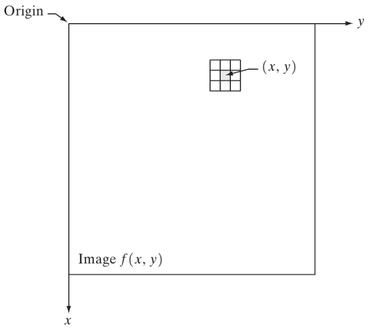


Agenda

1. Introduction and Definitions
2. Point (pixelwise) operations
3. Slicing grey levels
4. Image Histogram
   - Histogram equalisation
5. Filtering
   - Convolution
   - Smoothing filter
   - Order statistics
   - Non-local filtering
   - Sharpening

Space domain filtering

$$g(x, y) = T\left[f(x, y)\right],$$



Convolution

- Operation over a **neighborhood** of $f(x, y)$ generating a single value for pixel $g(x, y)$
- A filter $w()$ is designed for some purpose so that.

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t),$$

  - each x, y is computed so that $w()$ slides over all image f
  - the filter has size $m \times n$, with: $m = 2a + 1$ e $n = 2b + 1$.

Convolution

- The convolution can be represented by the $*$ operator:

$$w(x, y) * f(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t),$$

Convolution vs. cross-correlation

- The cross-correlation represents the sum of the point-wise products of the filter and image, centred at x, y

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t),$$

- Note that cross-correlation and convolution are equivalent if the filter is symmetric.

Vector representation

- A vector representation can be useful, writing the filtering as:

$$
\begin{aligned}
&= \mathbf{w}^T \mathbf{z} \\
&= \sum_{k=1}^{mn} w_k z_k \\
R &= w_1 z_1 + w_2 z_2 + \cdots + w_{mn} z_{mn},
\end{aligned}
$$

R is the response of the filter w centred in a given pixel and its neighbours $\mathbf{z}$

Mean

- **Mean**:

$$w(x, y) = \frac{1}{mn},$$

- Properties:

$$N = \sum_i (a - a_i)^2$$

$$a = \bar{a}_i$$

- The mean filter minimises the squared error in the neighborhood

Gaussian filter

$$G_{1D}(x,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

$$G_{2D}(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$G_{ND}(\vec{x},\sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^N} e^{-\frac{|\vec{x}|^N}{2\sigma^2}}$$

σ is the standard deviation of a Gaussian distribution of zero mean.

- Also called Gaussian *kernel*, centred at the origin and considering equal variances/standard deviations for all dimensions.

Gaussian filter

- **2D Gaussian filter** (sampled version of the distribution):

$$G(x, \sigma) = w(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}},$$

  σ controls the diffusion or dispersion of the values

- Relationship with com heat transfer: each pixel value is a heat point, the variance/std codifies the diffusion time.
  - larger values of variance/diffusion will approach the mean.

Median

- **Median**:

$$w(x, y) = \text{median}(z_k | k = 1, ..., nm),$$

  z_k for $k = 1, ..., nm$ are neighbours of the pixel (x, y).
- it minimises the $L - 1$ norm, or the sum:

$$\sum_i |a - a_i|$$

  this error is more robust, and tend to avoid smoothing borders
- since it is a order statistic, it does not produces new values.

Other filters

- **Maximum**:
$$w(x, y) = \max(z_k | k = 1, ..., nm),$$
- **Minimum**:
$$w(x, y) = \min(z_k | k = 1, ..., nm),$$

Non Local Means

- Assuming we have many pixels p with the same value p_0, but with some additive noise n:
$$p = p_0 + n$$
- If n is a random variable, then each pixel is composed of a different realisation of n
$$p_1 = p_0 + n_1$$
$$p_2 = p_0 + n_2$$
$$\ldots$$
- Non Local Means searches for regions over all image (not only locally) with similar values, and computes the mean using all regions

Non Local Means

- The different approaches try to:
  1. find similar regions
  2. filter those values

B. Goossens, H.Q. Luong, A. Pizurica, W. Philips, "An improved non-local means algorithm for image denoising,"

2008 International Workshop on Local and Non-Local Approximation in Image Processing (LNLA2008)

Agenda

Sharpening and image derivative

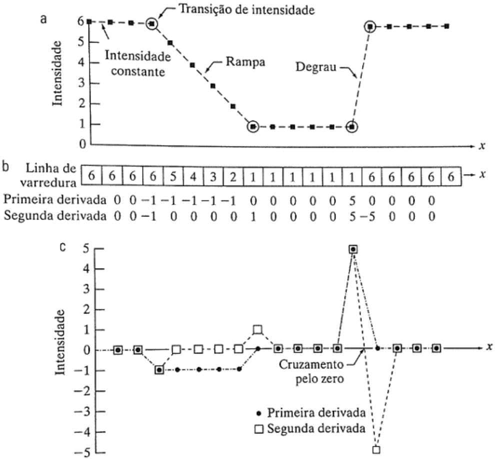
- The sharpening operation tries to enhance transitions of intensities.
- The derivatives are useful in this case since it codifies the transitions. For a given function $f(x)$ the partial derivative can be written as:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- The second order derivative:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

Sharpening and image derivative



Laplacian

- In 2d, the simplest isotropic operator is the Laplacian:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Which can be obtained via approximations with:

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

- For a filter 3×3:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Sharpening using the Laplacian filter

- We add the result of a Laplacian filter in the original image

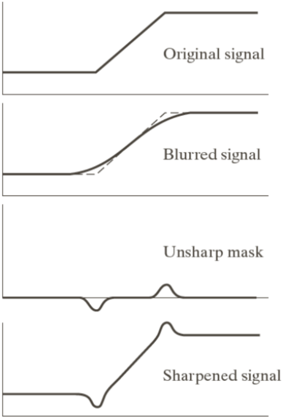
$$g(x, y) = f(x, y) + c|\nabla^2 f(x, y)|$$

- Some $c \leq 1$ will compensate the additive term,

Unsharp mask

1. Blur the original image
2. Subtract the blurred version from the original,
3. Add the matrix obtained in step (2) to the original image.

Unsharp mask



Bibliography I

GONZALEZ, R.C.; WOODS, R.E. ⋆
Processamento Digital de Imagens, 3.ed
Capítulo 3.
Pearson, 2010.