Complete Graph Averaging

Laurent Najman

October 15, 2013

1 Overview

In any digital image, the measurement of the three observed color values at each pixel is subject to some perturbations. These perturbations are due to the random nature of the photon counting process in each sensor. The noise can be amplified by digital corrections of the camera or by any image processing software. For example, tools removing blur from images or increasing the contrast enhance the noise.

The principle of the first denoising methods was quite simple: Replacing the color of a pixel with an average of the colors of nearby pixels. The variance law in probability theory ensures that if nine pixels are averaged, the noise standard deviation of the average is divided by three. Thus, if we can find for each pixel nine other pixels in the image with the same color (up to the fluctuations due to noise) one can divide the noise by three (and by four with 16 similar pixels, and so on). This looks promising, but where can these similar pixels be found?

The most similar pixels to a given pixel have no reason to be close at all. Think of the periodic patterns, or the elongated edges which appear in most images. It is therefore licit to scan a vast portion of the image in search of all the pixels that really resemble the pixel one wants to denoise. Denoising is then done by computing the average color of these most resembling pixels. The resemblance is evaluated by comparing a whole window around each pixel, and not just the color. Intuitively, it amounts at building a complete edge-weighted graph linking all image pixels, the data of a given pixel being all the pixels in a neighborood of the pixel under study[1]. This new filter is called complete-graph averaging and in a continuous form, it writes:

$$CGA(u(p)) = \frac{1}{C(p)} \int f(d(B(p), B(q))\, u(q)\, dq \tag{1}$$

where $d(B(p), B(q))$ is an Euclidean distance between image windows centered respectively at p and q, f is a decreasing function and $C(p)$ is the normalizing factor.

[1]However, we will not build the complete graph, but we will implement it using the techniques seen in the programming sessions of the course

2 Pixelwise Implementation

The denoising of a color image $u = (u_1, u_2, u_3)$ at a certain pixel p writes

$$\hat{u_i}(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u_i(q)\, w(p,q), \qquad C(p) = \sum_{q \in B(p,r)} w(p,q) \tag{2}$$

where $i = 1, 2, 3$ and $B(p, r)$ indicates a neighborhood centered at p and size $(2r + 1) \times (2r + 1)$ pixels. This research zone is limited to a square neighborhood of fixed size because of computation restrictions.

The weight $w(p, q)$ depends on the squared Euclidean distance $d^2 = d^2(B(p, f), B(q, f))$ of the $2f + 12f + 1$ color windows centered respectively at p and q.

$$d^2(B(p,f), B(q,f)) = \frac{1}{3(2f+1)^2} \sum_{i=1}^{3} \sum_{j \in B(0,f)} (u_i(p+j) - u_i(q+j))^2. \tag{3}$$

That is, each pixel value is restored as an average of the most resembling pixels, where this resemblance is computed in the color image. So for each pixel, each channel value is the result of the average of the same pixels.

We use an exponential kernel in order to compute the weights $w(p, q)$

$$w(p,q) = e^{-\frac{max(d^2 - 2\sigma^2, 0.0)}{h^2}} \tag{4}$$

where σ denotes the standard deviation of the noise and h is the filtering parameter set depending on the value of σ. The weight function is set in order to average similar windows up to noise. That is, windows with square distances smaller than $2\sigma^2$ are set to 1, while larger distances decrease rapidly accordingly to the exponential kernel.

The weight of the reference pixel p in the average is set to the maximum of the weights in the neighborhood $B(p, r)$. This setting avoids the excessive weighting of the reference point in the average. Otherwise, $w(p, p)$ should be equal to 1 and a larger value of h would be necessary to ensure the noise reduction. By applying the above averaging procedure we recover a denoised value at each pixel p.

3 Window-wise Implementation

The denoising of a color image $u = (u_1, u_2, u_3)$ in a certain window $B = B(p, f)$ (centered at p and size $(2f + 1) \times (2f + 1)$) writes

$$\hat{B}_i = \frac{1}{C} \sum_{Q=Q(q,f) \in B(p,r)} u_i(Q)\, w(B,Q), \qquad C = \sum_{Q=Q(q,f) \in B(p,r)} w(B,Q) \tag{5}$$

where $i = 1, 2, 3$, $B(p, r)$ indicates a neighborhood centered at p and size $(2r + 1) \times (2r + 1)$ pixels and $w(B(p, f), B(q, f))$ has the same formulation than in the pixelwise implementation.

In this way, by applying the procedure for all windows in the image, we shall dispose of $N = (2f + 1)$ possible estimates for each pixel. These estimates can be finally averaged at each pixel location in order to build the final denoised image.

$$\hat{u}_i(p) = \frac{1}{N^2} \sum_{Q=Q(q,f)|q\in B(p,f)} \hat{Q}_i(p) \tag{6}$$

4 Parameters

One of the important parameter is the size of the research window. This is a 21x21 window for small and moderate values of σ. The size of the research zone is increased to $35x35$ for large values of σ due to the necessity of finding more similar pixels to reduce further the noise.

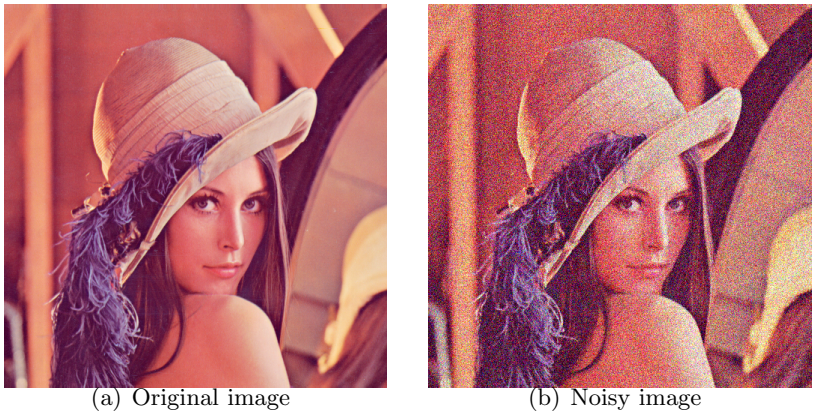
The value of the filtering parameter writes $h = k\sigma$. The value of k decreases as the size of the patch increases. For larger sizes, the distance of two pure noise patches concentrates more around $2\sigma^2$ and therefore a smaller value of k can be used for filtering.

Parameters for grayscale images			
σ	Window Size	Research Window Size	h
$0 < \sigma \leq 15$	3x3	21x21	0.40σ
$15 < \sigma \leq 30$	5x5	21x21	0.40σ
$30 < \sigma \leq 45$	7x7	35x35	0.35σ
$45 < \sigma \leq 75$	9x9	35x35	0.35σ
$75 < \sigma \leq 100$	11x11	35x35	0.30σ

Parameters for color images			
σ	Window Size	Research Window Size	h
$0 < \sigma \leq 25$	3x3	21x21	0.55σ
$25 < \sigma \leq 55$	5x5	35x35	0.40σ
$55 < \sigma \leq 100$	7x7	35x35	0.35σ

5 Examples

Fig. 1 provides some examples of CGA denoising.



(a) Original image (b) Noisy image

(c) $\sigma = 10$ (d) $\sigma = 40$

Figure 1: An image, its noisy version and two different CGA with different parameters

6 Questions

A novel version of the pinkdev library is available on the website. It allows to read ppm images (color images). This can help you for the question 4. A set of images is available for testing (*i.e*, with the result of the CGA denoising).

1. Program a grayscale average filter (*i.e.*, just the mean in a small window around the pixel, as seen in the course).
2. Program a grayscale pixelwise CGA filter.
3. Program a grayscale windowwise CGA filter.
4. Extend to color images (3 pgm as input, R, G, B).
5. What are the main differences between the three versions (ie, classical average filter, pixelwise and window-wise) in term of both image quality, PSNR and preservation of details? You may want to program a PSNR estimator to answer that question, but it does not answer to the qualitative part. For information about PSNR, Wikipedia is a good start: `http://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio`