CS664 Lecture 1: Image Restoration/Denoising

Scribed by Ozan Hafizogullari

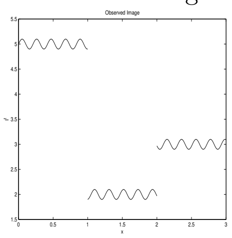
27 January 2000

When an image is taken by a camera, light photons fall through the camera lens onto one of the cells in an array of charged coupled device. These devises transform light energy of the photons into electrical charge. The charge pattern stored in the CCD array is then read out sequentially into an analog, voltage-modulated signal. This signal is converted into the raw image by discretizing it.

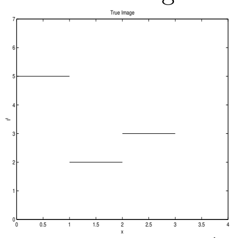
This process is very error-prone. Even two images of a stationary scenery taken by the same camera differ due to noise. *Image restoration problem* is recovering the uncontaminated image from the observed image. This can be summarized by the following:

$$\textbf{T}\text{rue Image} + \textbf{N}\text{oise} = \textbf{O}\text{bserved Image}$$

Observed Image



True Image



From now on, we will use I to stand for the observed image and I^t for the true image. To solve the image restoration problem, we make the following assumptions:

1. I is similar to I^t.
2. I^t is "smooth". (Spatial coherence)

One solution to this problem called *filtering* works as follows:

Let in[] be the observed picture and r a positive integer. Then, a solution to the restoration problem is

$$\text{out}[x] = \text{average}(\text{in}[x-r], \text{in}[x-r+1], \ldots, \text{in}[x+r])$$

When, as above, the operation is *average*, this scheme is called *box filtering*. When the operation is *median*, it is called *median filtering*. The interval $[x-r, x+r]$ with length $d = 2r+1$ is called a *window*.

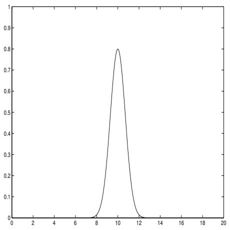
The obvious algorithm for box filtering is $O(nd)$ where n is the length of the array. However, $O(n)$ time with a small factor is possible using dynamic programming:

$$\text{out}[x] = \sum_{i=d-r}^{d+r} I[i]/d = \text{out}[x-1] + (I[d+r+1] - I[d-r])/d$$

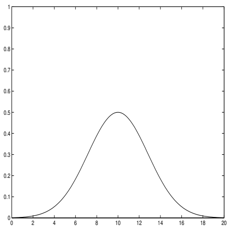
The image restoration problem is an instance of a more general family of problems called *labelling problems*. In this family, we are given a picture P, s.t. $P \subseteq \mathcal{Z} \times \mathcal{Z}$. Each pixel p, s.t. $p \in P$ has an observed intensity $I(p)$ and a true intensity $I^t(p)$. The aim is to find a labelling f, s.t. $f : P \to L$, where L is the set of estimated intensities.

In this scheme, the relationship between I^t and I is important, esp. $\Pr(I(p) = a | I^t(p) = b)$. Suppose $I^t(p) = 10$. What do we observe about the distribution of the intensity of the pixel p, if we make many experiments?

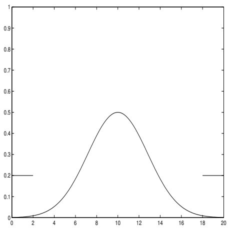
Good Camera



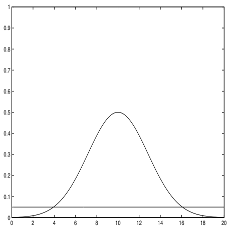
Bad Camera



Salt and Pepper Noise



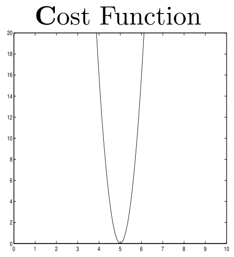
Contamination Model



Formalization of this idea requires a *cost* function $C(p, f(p))$, based on the observed data $I(p)$ and noise model. The intuition behind $C(p, f(p))$ is the cost of assigning $f(p)$ to the pixel p. There are several constraints on $C()$:

1. $C(p, f(p)) \geq 0$, for all p and $f(p)$.
2. $C(p, I(p)) = 0$

3. The cost should increase as we $f(p)$ gets further away from $I(p)$ and the resulting graph should be symmetric.



Common cost functions are $(f(p) - I(p))^2$ and $|f(p) - I(p)|$. $I(p) - f(p)$ is called the *residual*.

Given a cost function, we can use the following algorithm to find a solution to the image restoration problem:

1. At each pixel, look in a local window $W(p)$.
2. Assume all the pixels in the window will get the same label.
3. Compute the best label for p:

$$\hat{\text{f}}(p) = \min_{l \in L}\Big(\sum_{p' \in W(p)} C(p', l)\Big)$$

Fact 1 *Given $\{x_i\}$, the z that minimizes $\sum_i (z - x_i)^2$ is the average of x_i.*

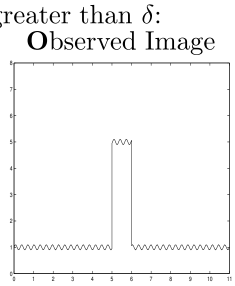
Proof: Take the derivative of $\sum_i (z - x_i)^2$ and set it to zero.

Hence, if the cost function is $(f(p) - I(p))^2$, we have $\hat{\text{f}}(p)$ is the average of the intensities in the local window.

Fact 2 *Given $\{x_i\}$, the z that minimizes $\sum_i (z - x_i)^2$ is the median of x_i.*

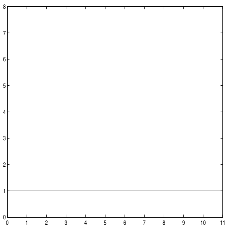
Hence, if the cost function is $|f(p) - I(p)|$, then $\hat{\text{f}}(p)$ is the median of the intensities in the local window.

Discontinuities create problems in vision. Suppose the observed picture is as follows and further assume that the width of the local window we are using is much greater than δ:



This will cause the box filtering method to give the following output:

Output



This is a common problem in many vision algorithms. In general vision algorithms make "thin" objects vanish.

So, if we compare box filtering with median filtering, we see that median filtering is better behaved at the boundaries. However, box filtering is faster through dynamic programming and more importantly it is "linear".

The definition of "linear" requires us to use some ideas from the theory of linear systems.

Definition 1 *Let I be a function from locations in space to a scalar value. Let B be a function on the set of such functions: $I[x] \to B(I)[x]$. B is* linear *iff*

$$\forall \textit{ scalar } \alpha,\ \beta \textit{ and } I,\ I'.B(\alpha I + \beta I')[x] = \alpha B(I)[x] + \beta B(I')[x]$$

This is equivalent to the conjunction of:

1. *$B(\alpha I) = \alpha B(I)$, for all α.*
2. *$B(I + I') = B(I) + B(I')$, for all I and I'.*

Another important definition is *shift invariance.* Shifting is defined by

$$(\mathrm{shift}_{\delta x, \delta y} I)[x, y] = I[x - \delta x, y - \delta y]$$

Definition 2 *B is* shift-invariant *iff $B(\textit{shift}_\triangle I) = \textit{shift}_\triangle B(I)$*

Any uniform, local algorithm is shift-invariant. Box filtering is, in addition, linear.