MRT Image Processing I Report

Suketu Patni, 23B1299

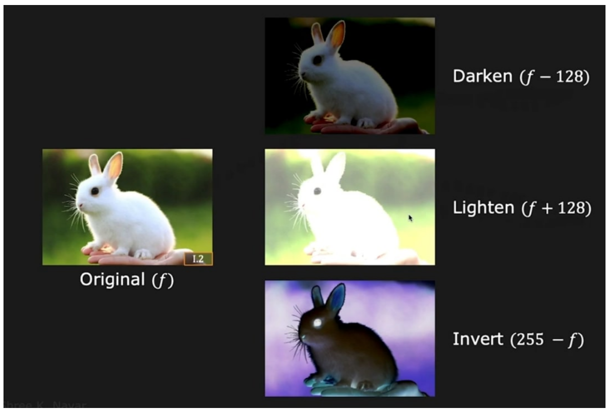
2 January, 2024

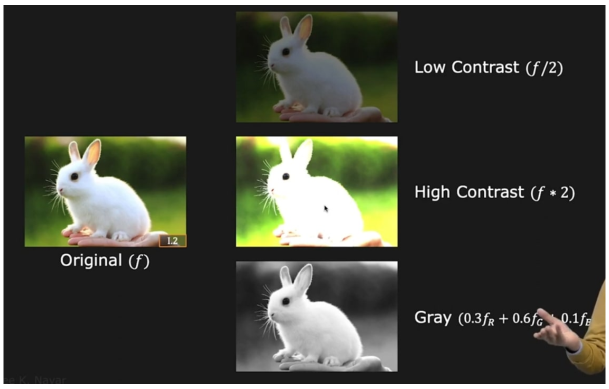
1 Motivation

- Link to playlist
- Transform image to one that is clearer or easier to analyze.
- Images taken at night may be grainy due to less light.
- Removing motion-blur.
- Removing defocus and making image crisper.
- Detecting edges/corners etc.

2 Pixel (Point) Processing

1. An black and white image is simply a function that maps pixel at (x, y) to an intensity $f(x, y)$.
2. For coloured images we may have $r(x, y), g(x, y), b(x, y)$.
3. A simple transformation is $g(x, y) = T(f(x, y))$ where T is a function that maps one intensity to an other (pixel transform does not depend on pixel location).





3 LSIS (Linear Shift Invariant Systems)

1. $f(x) \to LSIS \to g(x)$
2. Linearity: If $f_1 \to LSIS \to g_1$ and $f_2 \to LSIS \to g_2$ then $\alpha f_1 + \beta f_2 \to LSIS \to \alpha g_1 + \beta g_2$.
3. Shift Invariance: If $f(x) \to LSIS \to g(x)$ then $f(x - a) \to LSIS \to g(x - a)$
4. An ideal lens may be considered an LSIS. If g is defocused image and f is focused image, linearity is implied by "brighter the scene, both f and g are that much brighter", and shift invariance is implied by "shifting the scene parallel to the lens shifts both f and g by that amount" (ignore any magnification for now).

4 Convolutions

1. If f convolved with h yields g, then

$$g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)h(x-\tau)\, d\tau$$

2. The convolve operator $*$ is commutative and associative.

3. Link for convolution demo

4. System is an LSIS $\iff$ System is performing a convolution.

5. It is easily shown that $f_1 * h = g_1$ and $f_2 * h = g_2 \Rightarrow (\alpha f_1 + \beta f_2) * h = \alpha g_1 + \beta g_2$ (linearity) and $f(x) * h(x) = g(x) \Rightarrow f(x-a) * h(x) = g(x-a)$.

6. To find h, convolve the unit impulse function $\delta(x)$ with the LSIS. $\delta(x)$ is the identity under the $*$ operator.

$$\delta(x) = \lim_{\epsilon \to 0} \delta_\epsilon(x)$$

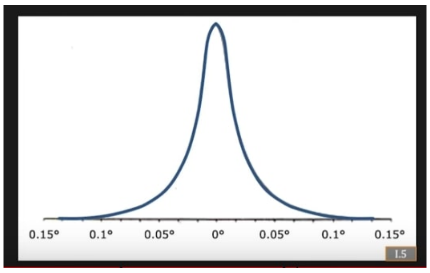
where

$$\delta_\epsilon(x) = \left\{ \begin{array}{c} \frac{1}{2\epsilon}, |x| \leq \epsilon \\ 0, |x| > \epsilon \end{array} \right\} \tag{1}$$

7. $\delta(x)$ is zero at every nonzero real value yet still achieves unity as its integral over the real line.

8. Thus, $h(x)$ is called the impulse response of the LSIS.

9. Impulse response of human eye: Since the human eye is a lens, it is an LSIS. The unit impulse function is 2d (a distant bright star may be used). The impulse response for human eyes is often called the point spread function (PSF) and it is measured in degrees since retina is curved.



10. Cascaded systems: $f \to (h_1) \to (h_2) \to g$ is equivalent to $f \to (h_1 * h_2) \to g$.

11. 2D Convolution:

$$g(x,y) = f(x,y) * h(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau,\mu)h(x-\tau, y-\mu)\, d\tau d\mu$$

5 Linear Image Filters

1. Convolution with discrete images:

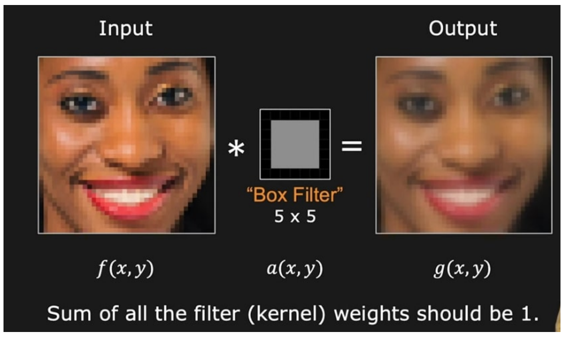
$$f[i,j] \to h[i,j] \to g[i,j] \Rightarrow g[i,j] = \sum_{m=1}^{M} \sum_{n=1}^{N} f[m,n]h[i-m, j-n]$$

where image size is (M,N).

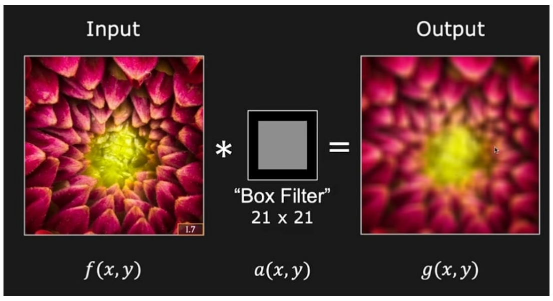
2. Here $h[i,j]$ is called a mask, kernel or a filter. We flip h along x, then along y, put centre of h on the pixel (i, j) we wish to find g of, then do a sum of products of pixel values of h and those of f for all pixels in h; then that sum is the pixel value of pixel at (i, j). We do this for all pixels in image.

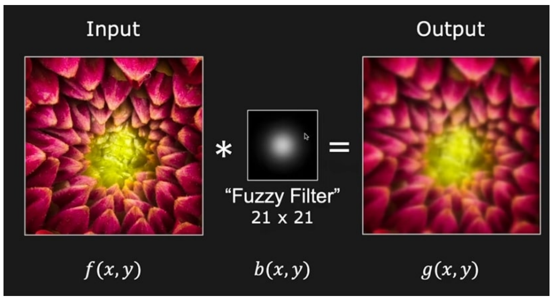
3. Border Issues: Solutions are either (a) ignore border pixels (don't convolve all of h for borders), (b) pad with a certain constant value, or (c) pad with reflected value of pixels nearby borders.

4. Examples: Box Filter (smooths images)



5. Fuzzy filters: Images smoothed with box filters have block-y regions of pixels. So we use fuzzy filters.





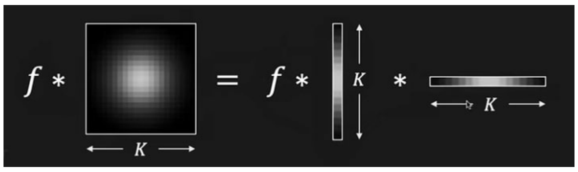
6. Fuzzy filters are formalized using Gaussian Kernels. Where σ^2 is the variance,

$$n_\sigma[i,j] = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{i^2+j^2}{\sigma^2}\right)} \tag{2}$$

where i, j are measured from the centre of the kernel h.

7. For standard deviation σ, put kernel size $K \approx 2\pi\sigma$. Greater the σ, more the blurring.

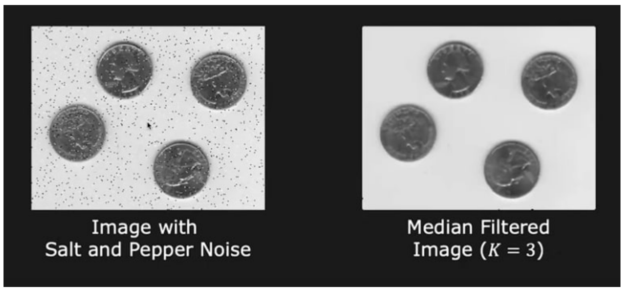
8. Gaussian smoothing is separable i.e.



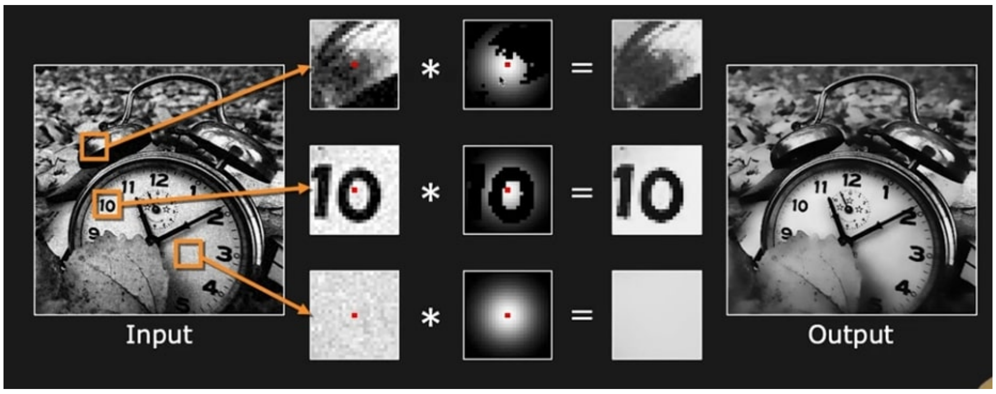
9. In fact the RHS requires only $O(K)$ multiplications while the LHS requires $O(K^2)$ multiplications. This separability property makes computation cheaper.

6 Non-Linear Image Filters

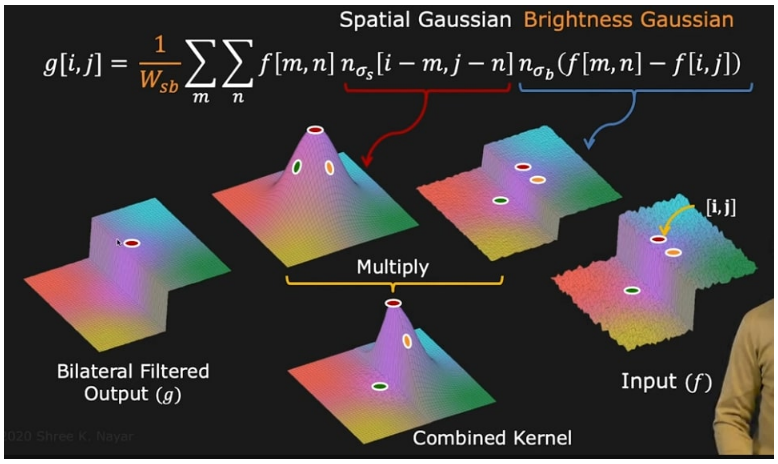
1. These filters take a more algorithmic approach (cannot be implemented using convolution).
2. Problem with smoothing is that it does not remove outliers, and it smooths edges and smears noise which increases blur.
3. Median Filtering: Sort the K^2 values in window centered at pixel, and assign the middle value to that pixel. Removes salt and pepper noise very cleanly (but with some edge blurring). Larger K causes blurring of image detail.



4. When noise is not simple salt and pepper noise, the background may get textured (so it's not effective).
5. Bilateral filtering: Bias the Gaussian kernel such that pixels not similar in intensity to center pixel receive lower weight (even 0). Blur only similar pixels.



6. Noise is removed much more cleanly. This is similar to designing a custom Gaussian for each pixel.
7. Whereas the regular Gaussian would smooth out both the noise and the edges, the bilateral filter retains edges and still manages to remove noise.


Spatial Gaussian
Brightness Gaussian
$g[i,j] = \frac{1}{W_{sb}} \sum_m \sum_n f[m,n]\, n_{\sigma_s}[i-m, j-n]\, n_{\sigma_b}(f[m,n] - f[i,j])$
[i, j]
Multiply
Bilateral Filtered Output (g)
Input (f)
Combined Kernel
2020 Shree K. Nayar

8. It may be constructed as

$$g[i,j] = \frac{1}{W_{sb}} \sum_m \sum_n f[m,n] n_{\sigma_s}[i-m, j-n] n_{\sigma_b}(f[m,n] - f[i,j]) \tag{3}$$

where

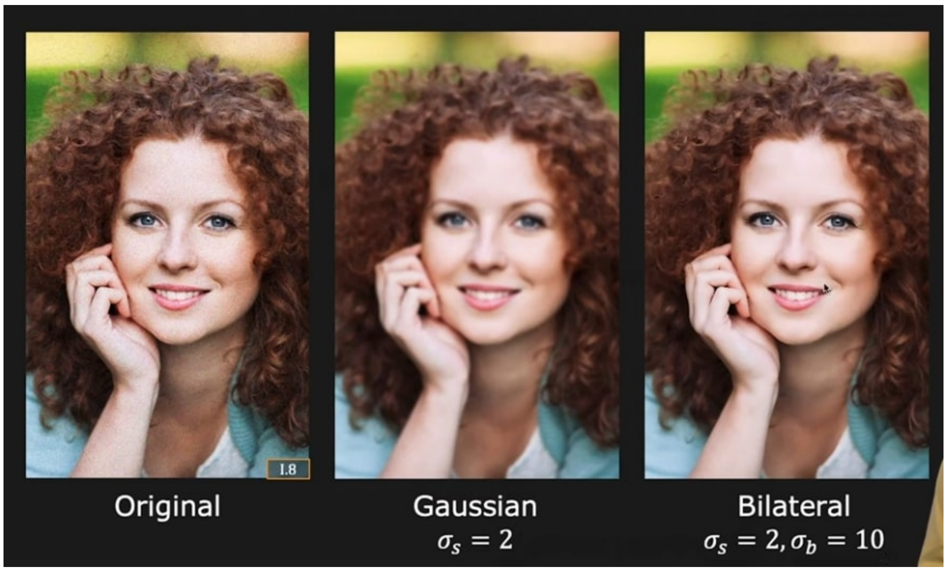
$$n_{\sigma_s}[m,n] = \frac{1}{2\pi\sigma_s^2} e^{-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma^2}\right)} \tag{4}$$

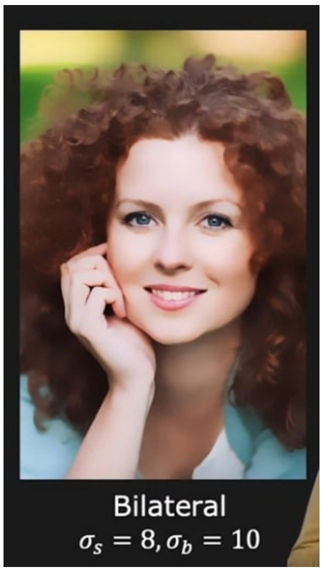
$$n_{\sigma_b}(k) = \frac{1}{\sqrt{2\pi}\sigma_b} e^{-\frac{1}{2}\left(\frac{k^2}{\sigma_b^2}\right)} \tag{5}$$

and

$$W_{sb} = \sum_m \sum_n n_{\sigma_s}[i-m, j-n] n_{\sigma_b}(f[m,n] - f[i,j]) \tag{6}$$

9. In the bilateral filter, as σ_s gets larger, we achieve a water-colour-y effect, whereas if σ_b gets larger, it turns into a regular Gaussian filter.





7 Template Matching by Correlation

1. The problem is to identify a small pattern (template) in a large image. We slide the template across, and whenever the squared sum of differences between it and the image pixels is lowest, we return a match i.e. we minimize

$$E[i,j] = \sum_m \sum_n (f[m,n] - t[m-i,n-j])^2 \tag{7}$$

   over all i, j i.e. maximize

$$R_{tf}[i,j] = sum_m \sum_n f[m,n]t[m-i,n-j] \tag{8}$$

   as the sums of squares of picture and template values are constant.

2. This quantity R_{tf}, also denoted as $t \bigotimes f$, is called the cross-correlation function.
3. This is very similar to a convolution of template with image except that correlation doesn't involve any flipping.
4. The problem is that if values of $f[m,n]$ are large, the correlation might be high even if value are not similar to those in the template. So we normalize it as

$$N_{tf}[i,j] = \frac{R_{tf}[i,j]}{\sqrt{sum_m \sum_n f^2[m,n]}\sqrt{sum_m \sum_n t^2[m-i,n-j]}} \tag{9}$$

   to account for energy differences, and then find the pixel (i, j) for which this is maximized.