

Image filtering



Images by [Pawan Sinha](#)

What is an image?

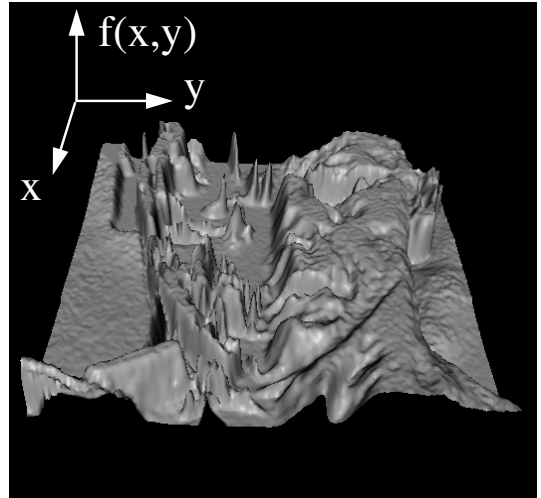
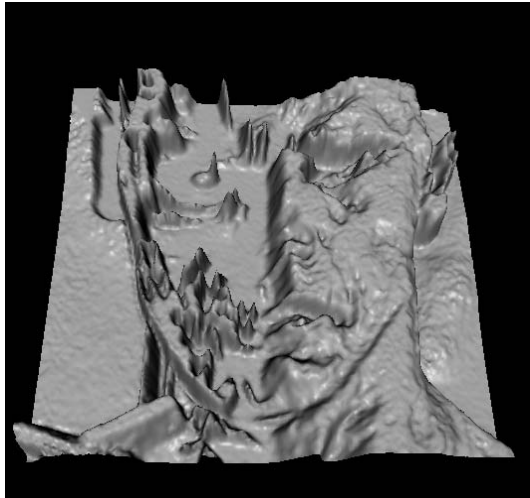
We can think of an **image** as a function, f , from \mathbb{R}^2 to \mathbb{R} :

- ◆ $f(x, y)$ gives the **intensity** at position (x, y)
- ◆ Realistically, we expect the image only to be defined over a rectangle, with a finite range:
 - $f: [a,b] \times [c,d] \rightarrow [0,1]$

A color image is just three functions pasted together.
We can write this as a “vector-valued” function:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

Images as functions



What is a digital image?

In computer vision we usually operate on **digital (discrete)** images:

- ◆ **Sample** the 2D space on a regular grid
- ◆ **Quantize** each sample (round to nearest integer)

If our samples are Δ apart, we can write this as:

$$f[i, j] = \text{Quantize}\{ f(i \Delta, j \Delta) \}$$

The image can now be represented as a matrix of integer values

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Filtering Operations Use Masks

- Masks operate on a neighborhood of pixels.
- A mask of coefficients is centered on a pixel.
- The mask coefficients are multiplied by the pixel values in its neighborhood and the products are summed.
- The result goes into the corresponding pixel position in the output image.

36	36	36	36	36
36	36	45	45	45
36	45	45	45	54
36	45	54	54	54
45	45	54	54	54

Input Image

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

3x3 Mask

**	**	**	**	**
**	39	**	**	**
**	**	**	**	**
**	**	**	**	**
**	**	**	**	**

Output Image

Noise

Image processing is useful for noise reduction...



Original



Salt and pepper noise



Impulse noise



Gaussian noise

Common types of noise:

- ◆ **Salt and pepper noise:** contains random occurrences of black and white pixels
- ◆ **Impulse noise:** contains random occurrences of white pixels
- ◆ **Gaussian noise:** variations in intensity drawn from a Gaussian normal distribution

Practical noise reduction

How can we “smooth” away noise in a single image?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	100	130	110	120	110	0	0
0	0	0	110	90	100	90	100	0	0
0	0	0	130	100	90	130	110	0	0
0	0	0	120	100	130	110	120	0	0
0	0	0	90	110	80	120	100	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Mean filtering

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

Mean filtering

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Effect of mean filters

Gaussian
noise

Salt and pepper
noise

3x3



5x5



7x7



Cross-correlation filtering

Let's write this down as an equation. Assume the averaging window is $(2k+1) \times (2k+1)$:

$$G[i, j] = \frac{1}{(2k + 1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F[i + u, j + v]$$

We can generalize this idea by allowing different weights for different neighboring pixels:

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i + u, j + v]$$

This is called a **cross-correlation** operation and written:

$$G = H \otimes F$$

H is called the “filter,” “kernel,” or “mask.”

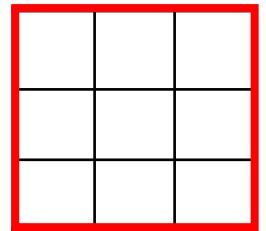
The above allows negative filter indices. When you implement need to use: $H[u+k, v+k]$ instead of $H[u, v]$

Mean kernel

What's the kernel for a 3x3 mean filter?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$



$H[u, v]$

Gaussian Filtering

A Gaussian kernel gives less weight to pixels further from the center of the window

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

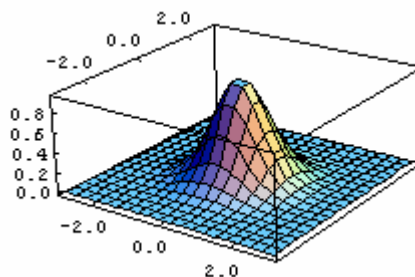
$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$H[u, v]$$

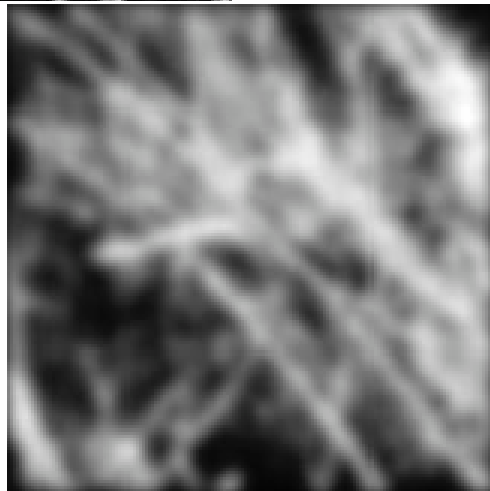
$$F[x, y]$$

This kernel is an approximation of a Gaussian function:

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Mean vs. Gaussian filtering



Convolution

A **convolution** operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i - u, j - v]$$

It is written: $G = H \star F$

Suppose H is a Gaussian or mean kernel. How does convolution differ from cross-correlation?

Median filters

A **Median Filter** operates over a window by selecting the median intensity in the window.

What advantage does a median filter have over a mean filter?

Is a median filter a kind of convolution?

Comparison: salt and pepper noise

Mean

Gaussian

Median

3x3



5x5



7x7



Comparison: Gaussian noise

Mean

Gaussian

Median

3x3



5x5



7x7

