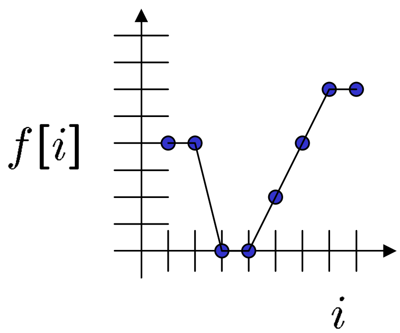
Linear image transforms

Let's start with a 1-D image (a "signal"): $f[i]$



$$f[i] = \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}$$

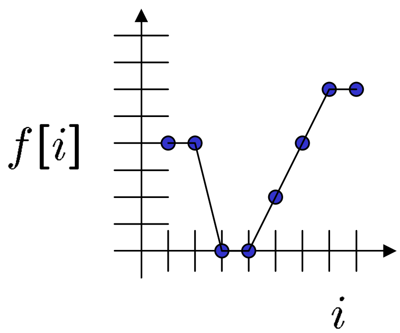
A very general and useful class of transforms are the **linear transforms** of f, defined by a matrix M

$$\underbrace{\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}}_{M[i,j]} \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \end{bmatrix}}_{f[i]} = \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \end{bmatrix}}_{g[i]}$$

$$g[i] = \sum_{j=1} M[i,j] f[j]$$

Linear image transforms

Let's start with a 1-D image (a "signal"): $f[i]$

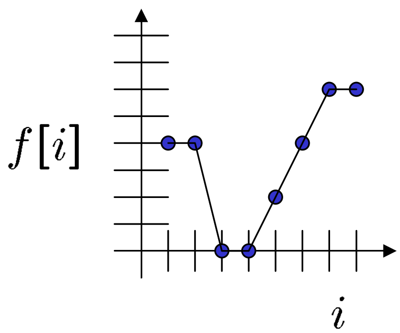


$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad f[i] \longrightarrow$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad f[i] \longrightarrow$$

Linear image transforms

Let's start with a 1-D image (a "signal"): $f[i]$



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad f[i] \rightarrow$$

$$\frac{1}{2}\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \qquad f[i] \rightarrow$$

Linear shift-invariant filters

$$\begin{bmatrix} * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ a & b & c & 0 & 0 & 0 & 0 & 0 \\ 0 & a & b & c & 0 & 0 & 0 & 0 \\ 0 & 0 & a & b & c & 0 & 0 & 0 \\ 0 & 0 & 0 & a & b & c & 0 & 0 \\ 0 & 0 & 0 & 0 & a & b & c & 0 \\ 0 & 0 & 0 & 0 & 0 & a & b & c \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * \end{bmatrix}$$

This pattern is very common
- same entries in each row
- all non-zero entries near the diagonal

It is known as a **linear shift-invariant filter** and is represented by a **kernel** (or **mask**) h:

$$h[i] = [a \quad b \quad c]$$

and can be written (for kernel of size 2k+1) as:

$$g[i] = \sum_{u=-k}^{-k} h[u] f[i+u]$$

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

2D linear transforms

We can do the same thing for 2D images by concatenating all of the rows into one long vector:

$$\hat{f}[i] = f[\lfloor i/m \rfloor, i\%m]$$

$$\underbrace{\begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}}_{M[i,j]} \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \end{bmatrix}}_{\hat{f}[i]} = \underbrace{\begin{bmatrix} * \\ * \\ \vdots \\ * \end{bmatrix}}_{\hat{g}[i]}$$

2D filtering

A 2D image f[i,j] can be filtered by a 2D kernel h[u,v] to produce an output image g[i,j]:

$$g[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] f[i+u, j+v]$$

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

$$g = h \otimes f$$

h is called the "filter," "kernel," or "mask."

Noise

Filtering 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

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

$g[x,y]$

9

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

11

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

$f[x, y]$

$h[u, v]$

When can taking an un weighted mean be bad idea?

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

$$f[x, y]$$

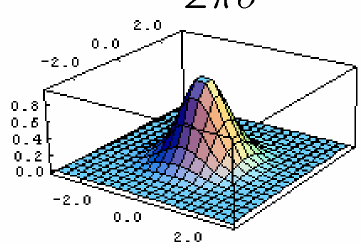
$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

$$h[u, v]$$

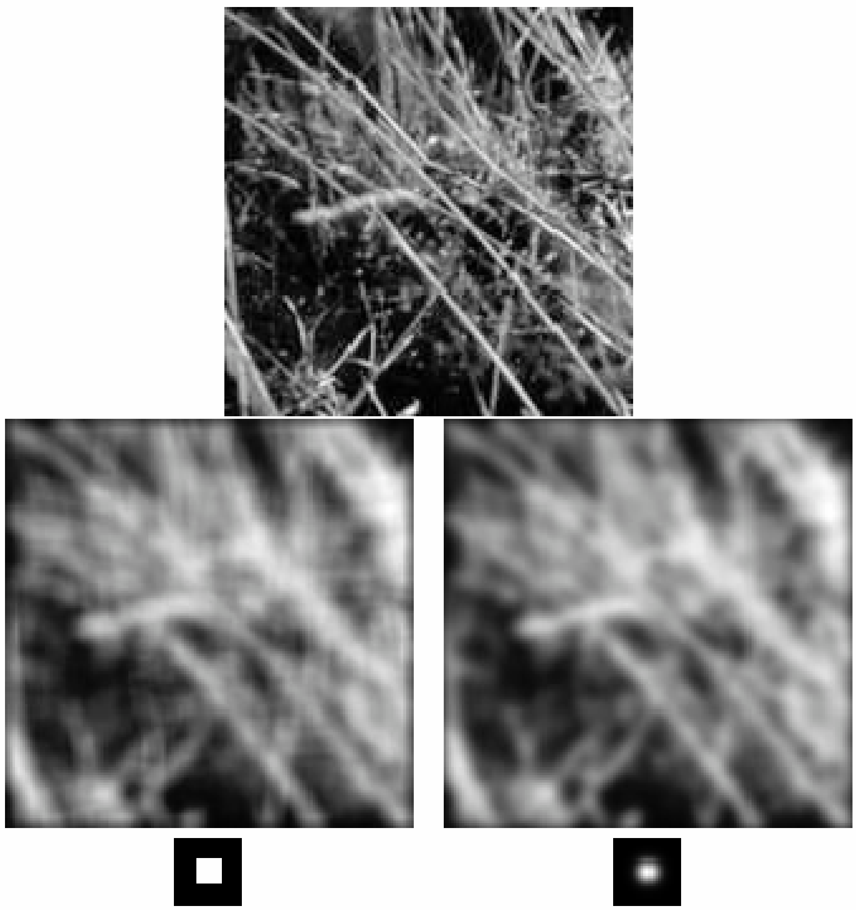
This kernel is an approximation of a Gaussian function:

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



13

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:

$$g[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] f[i-u, j-v]$$

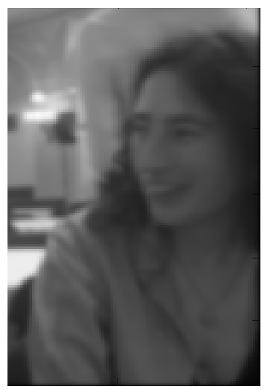
It is written: $g = h \star f$

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

Unsharp Masking

So, what does blurring take away?

-



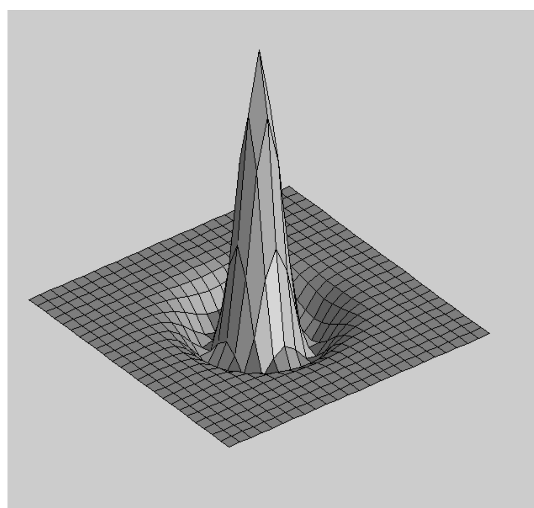
=

$+ \alpha$

=

Unsharp Masking (MATLAB)

```
Imrgb = imread('file.jpg');

im = im2double(rgb2gray(imrgb));

g= fspecial('gaussian', 25,4);

imblur = conv2(im,g,'same');

imagesc([im imblur])

imagesc([im im+.4*(im-imblur)])
```



```
logfilt = fspecial('log',25,4);
```

17

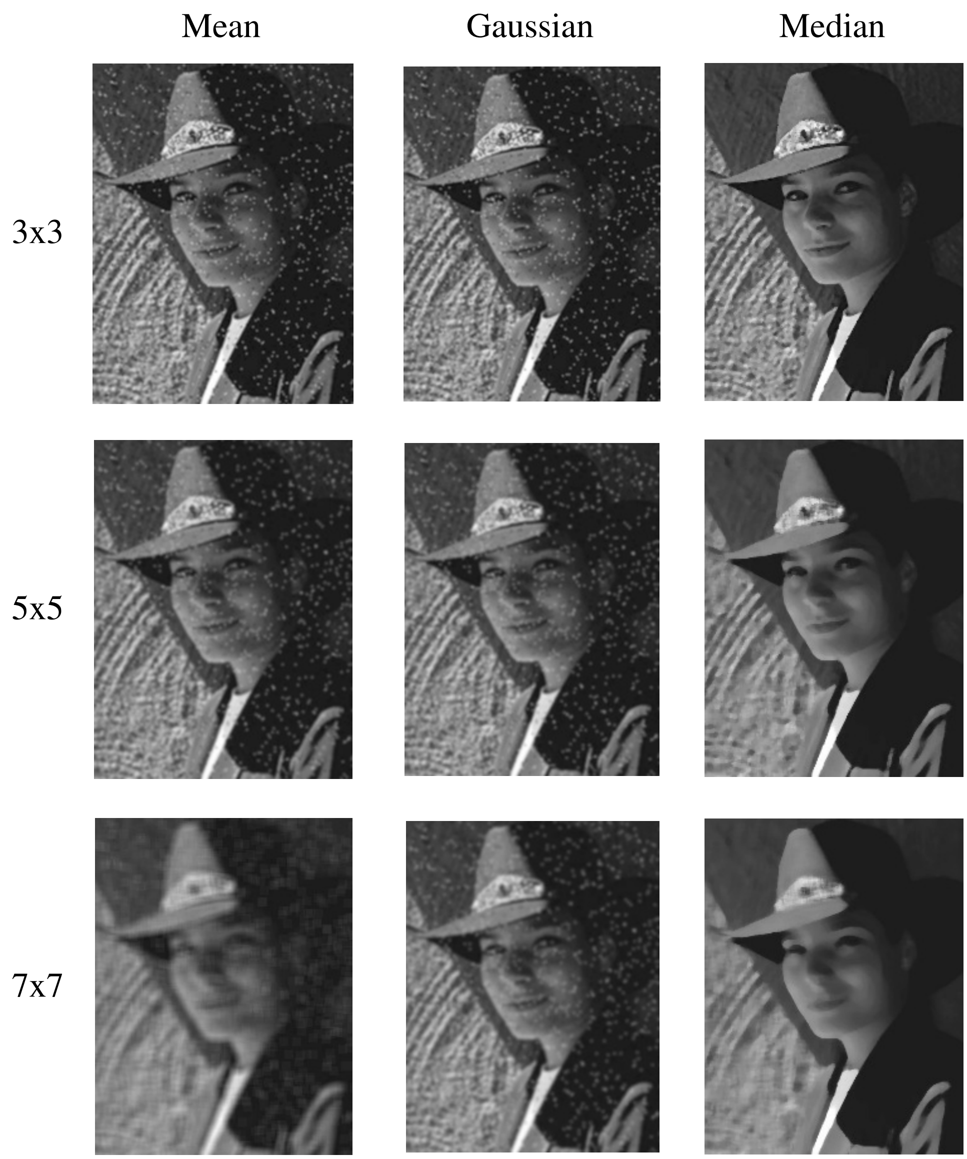
Median filters

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



19

Comparison: Gaussian noise

Mean | Gaussian | Median

3x3

5x5

7x7

20