OUTLINE

- Images:
  - Intensity histogram
  - Noise
- Image filtering:
  - Linear filter
    - Low-pass filtering
  - Non-linear filter
- 2D Fourier Transform

INTENSITY HISTOGRAM

The histogram is a graph showing the number of pixels in an image at each different intensity value found in that image.

Histogram $\quad h(r_k) = n_k$

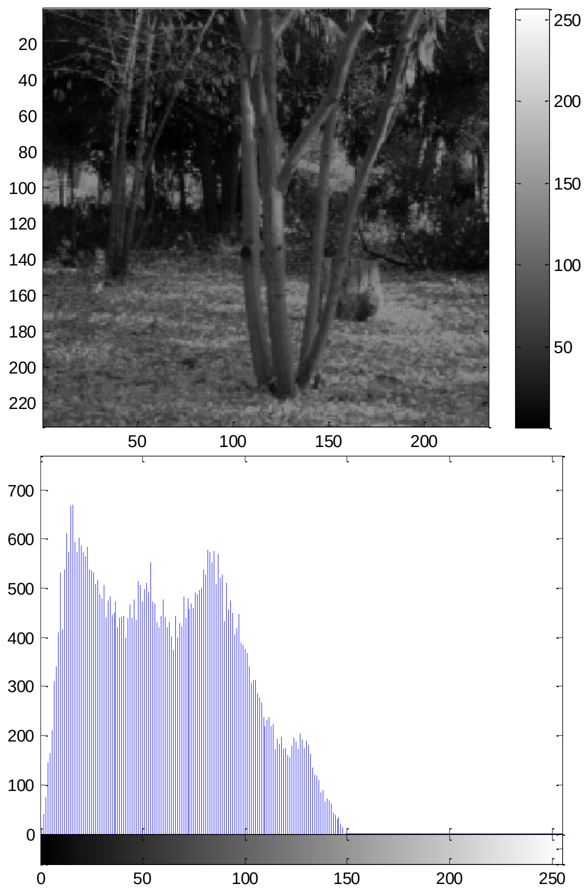
r_k is the k^{th} intensity value

n_k is the number of pixels in the image with intensity r_k

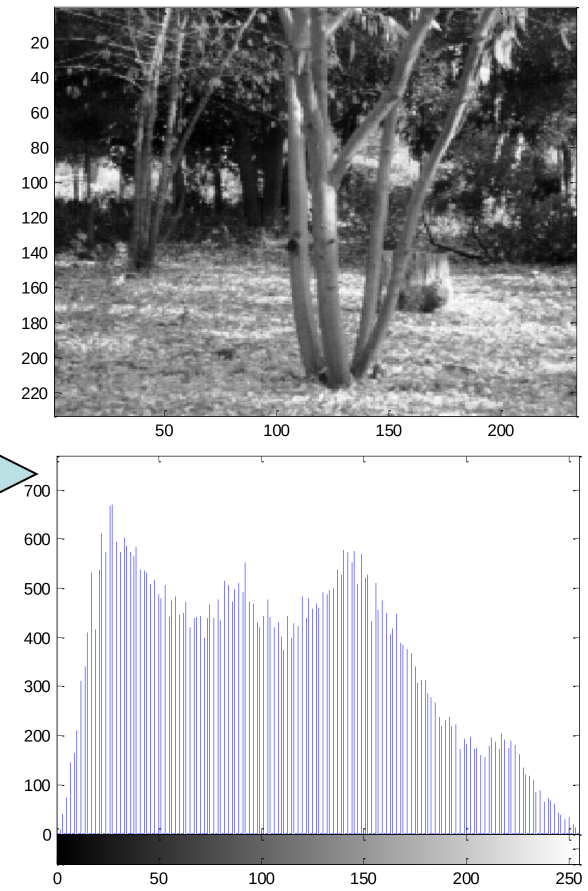
Contrast Stretching: expands the range (a_min, a_max) of intensity levels in an image a(i,j) so that it spans the full intensity range (omin, omax) of the recording medium or display device.

b(i ,j) = ((a(i ,j) –a_min)/(a_max-a_min)) (omax-omin)+omin

INTENSITY HISTOGRAM



Contrast Stretching



See histo.m

NOISE

- Digital image are corrupted by noise during image formation process (e.g light fluctuations, sensor noise, quantization effects, finite precision).

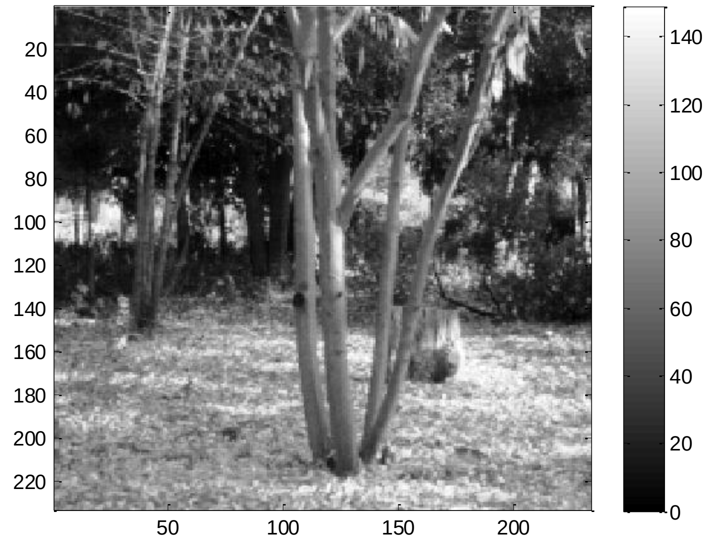
- Common types of noise (we often assume the noise is additive):
  – $I(x,y) = s(x,y) + ni$
  – Where $s(x,y)$ is the deterministic signal
  – ni is a random variable
  – Gaussian noise: variations in intensity drawn from a Gaussian normal distribution.

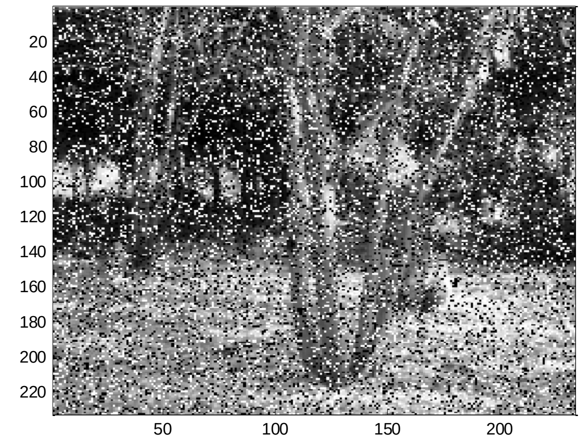
NOISE

  - Impulse (“shot”) noise, i.e. salt and pepper noise: contains random occurrences of black and white pixels.

- Digital images can be corrupted artificially in order to assess the performance of various vision processing algorithms.

NOISE



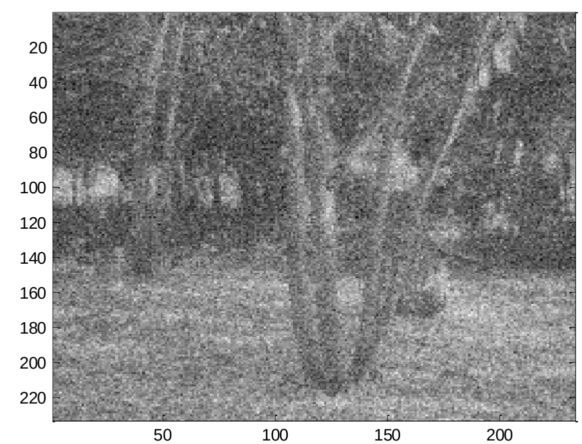


Salt and pepper noise

Noise density = 0.3

Gaussian noise

σ=10



σ=30

See noise.m

IMAGE PROCESSING

- Standard image processing operators map pixel values from one image to another:
- *Point operators*: where each output pixel's value only depends on the corresponding input pixel value (e.g., contrast adjustments and color transformations).
- *Neighborhood (area-based) operators*: where each new pixel's value depends on a small number of neighboring input pixel values (e.g. filtering).
- *Global operators* (e.g. histogram and Fourier Transform).

IMAGE (linear) FILTERING

- Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point of the image.

- Output is a linear function of the input.
- Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left).

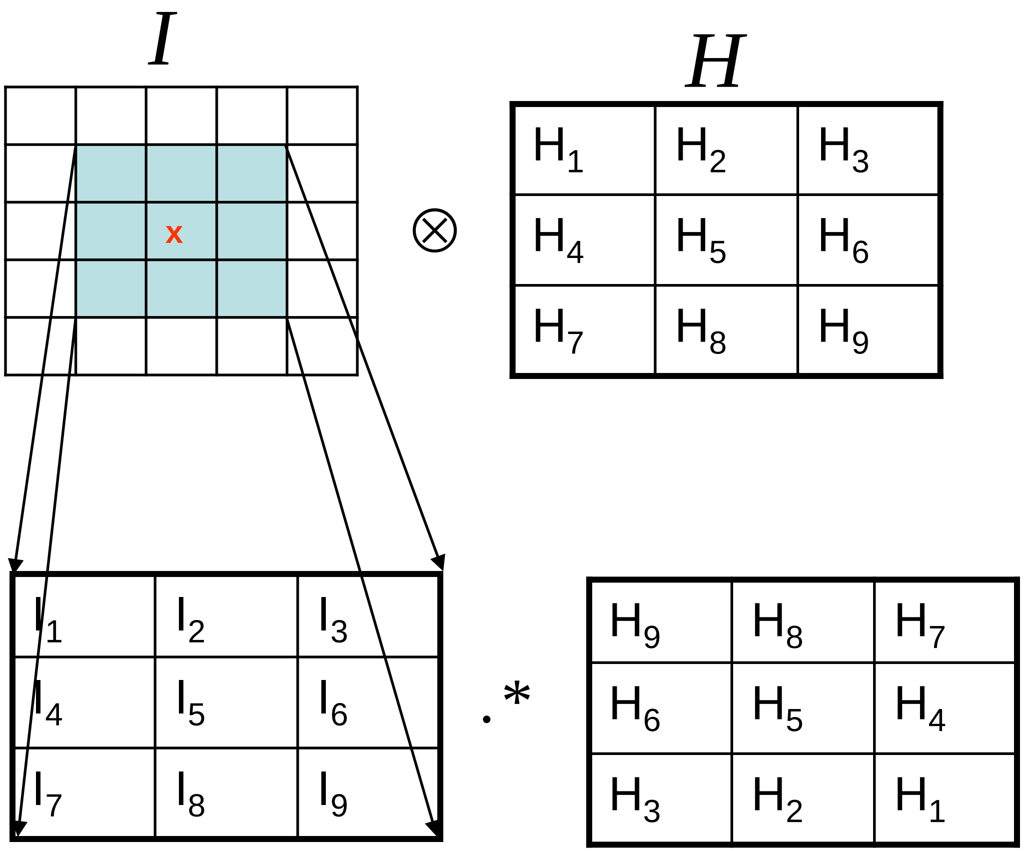
IMAGE FILTERING

- In order to form a new image whose pixels are a weighted sum of original pixel values, using the same set of weights (some function of a local neighborhood of the pixel) at each point:

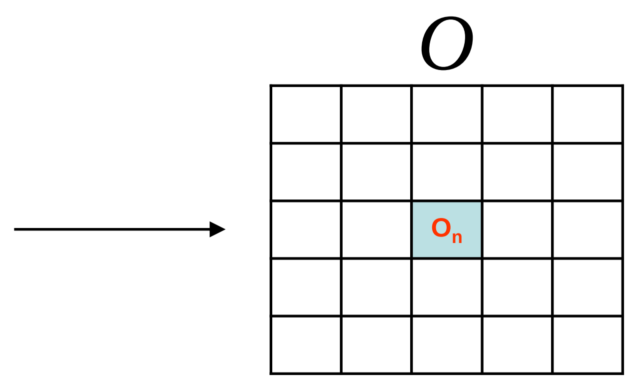
$$O(i, j) = I * H = \sum_k \sum_l I(k,l) H(i-k, j-l)$$

- where I is the input image, O is the output image and H is the convolution kernel (the two-dimensional spatial filter).

IMAGE FILTERING



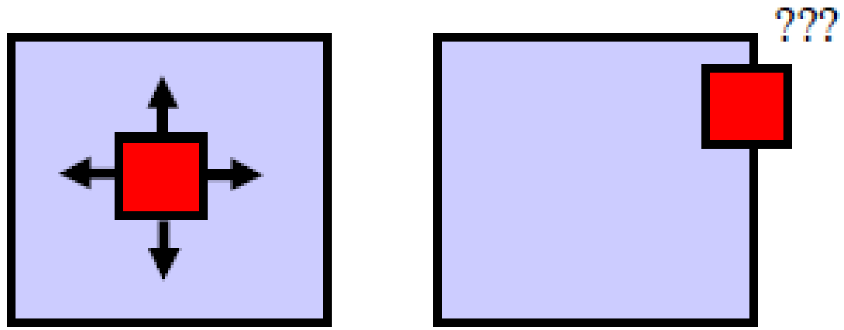
We don't want to only do this at a single pixel, of course, but want instead to "run the kernel over the whole image".



$$O_n = I_1H_9 + I_2H_8 + I_3H_7 + I_4H_6 + I_5H_5 + I_6H_4 + I_7H_3 + I_8H_2 + I_9H_1$$

IMAGE FILTERING

- Problem: what do we do for border pixels where the kernel does not completely overlap the image?



- Different border handling methods specify different ways of defining values for pixels that are off the image.
- One of the simplest methods is **zero-padding.**

IMAGE FILTERING

image

Origin $f(x, y)$

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

kernel

$w(x, y)$

1	2	3
4	5	6
7	8	9

Padded f

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	0	0	0	0
0	0	0	0	1	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	0	0	0	0	0	0	0	0

Rotated w

9	8	7	0	0	0	0	0	0
6	5	4	0	0	0	0	0	0
3	2	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	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	0	0	0	0	0	0	0	0

Full convolution result

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	1	2	3	0	0	0
0	0	0	4	5	6	0	0	0
0	0	0	7	8	9	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

Cropped convolution result

0	0	0	0	0
0	1	2	3	0
0	4	5	6	0
0	7	8	9	0
0	0	0	0	0

SMOOTHING

- Smoothing:
  - Smoothing filters are used for blurring and for noise reduction
  - Blurring is used in removal of small details and bridging of small gaps in lines or curves
  - Smoothing spatial filters include linear filters and nonlinear filters.

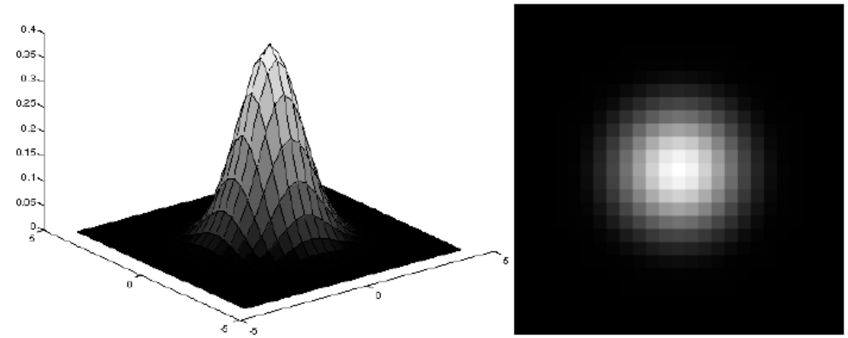
- Smoothing (linear filters):
  - by averaging (box filter) performs the average of pixels in a neighborhood;
  - with a Gaussian (low-pass filter) performs a weighted average of pixels in a neighborhood.

SMOOTHING

Box filter

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

since this is a linear operator, we can take the the average around each pixel by convolving the image with this 3x3 filter

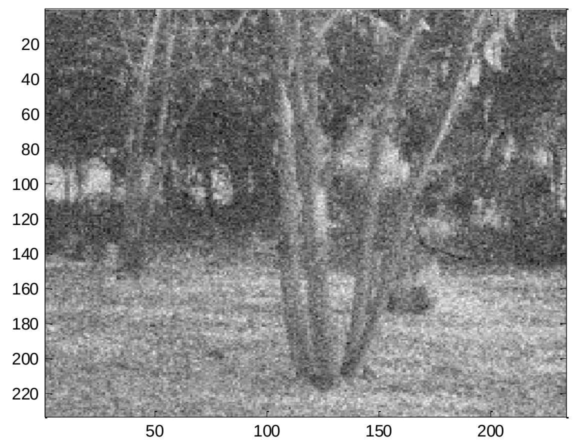


Gaussian Smoothing Filter (weighted averaging): the coefficients are a 2D Gaussian, i.e. it Gives more weight at the central pixels and less weights to the neighbors

$$G_\sigma \equiv \frac{1}{2\pi\sigma^2}\exp\{-\frac{x^2+y^2}{2\sigma^2}\}$$

IMAGE FILTERING: SMOOTHING

- Smoothing reduces noise (for appropriate noise models): since the pixels “are like” their neighbors, whereas the noise is independent from pixel to pixel.



σ=20

average

See noise_filtering.m

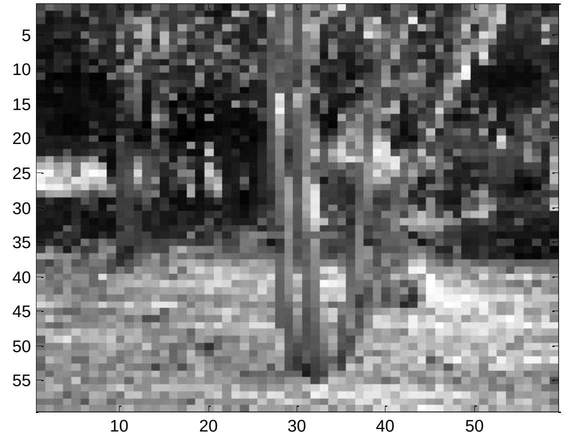
IMAGE FILTERING: Efficient Implementation

- Both, the Box filter and the Gaussian filter are separable:
  - First convolve each row with a 1D filter
  - Then convolve each column with a 1D filter.

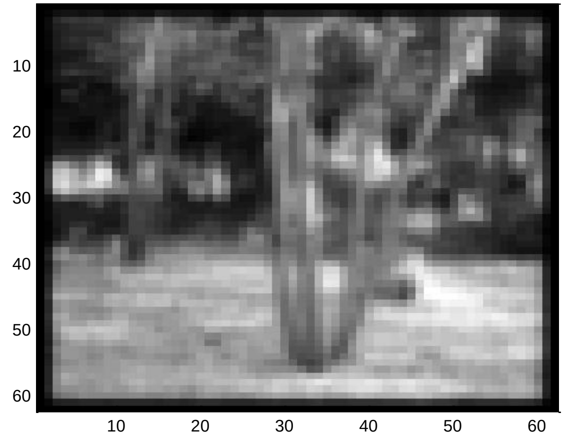
Separable Gaussian: **associativity**

$$G_\sigma * f = \left(g_{\sigma\rightarrow} * g_{\sigma\uparrow}\right) * f = g_{\sigma\rightarrow} * \left(g_{\sigma\uparrow} * f\right)$$

IMAGE RESOLUTION: SUBSAMPLING



Sub-sampling



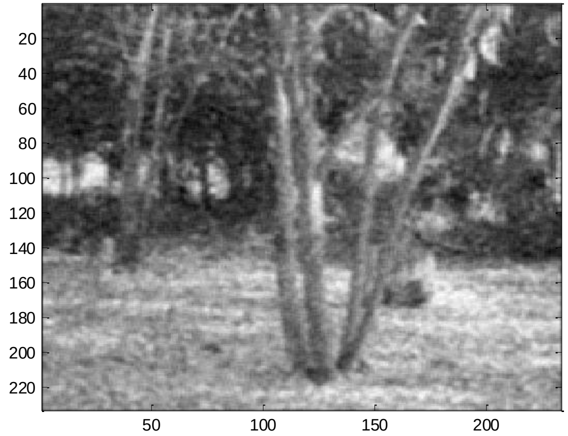
Low-pass filtering and sub-sampling

See subsampling.m

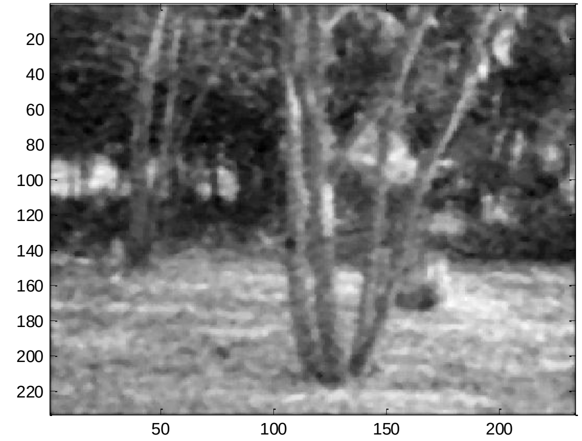
MEDIAN FILTER

- It is a non-linear filter: (1) rank-order neighborhood intensities and (2) take middle value.
  - median completely discards the spikes;
  - median preserves discontinuities;
  - median looses important details and produces patchy effect.

See noise_filtering.m



average



median

2D FOURIER TRANSFORM

- We define the Fourier transform of an image *g(x,y)* to be:

$$\Im\big(g(x,y)\big)(u,v)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}g(x,y)e^{-i2\pi(ux+vy)}dxdy$$

where the result is a complex valued function of *(u, v).*

- Any function that is not periodic can be expressed as the integral of sines and /or cosines multiplied by a weighing function

- The transform of an digital image is performed by FFT.

2D FOURIER TRANSFORM: 2D impulse

The impulse $\delta(x, y)$, $\qquad \delta(x, y) = \begin{cases} 1 & \text{if } x = y = 0 \\ 0 & \text{otherwise} \end{cases}$

The sifting property

$$\sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(x, y)\delta(x, y) = f(0,0)$$

and

$$\sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(x, y)\delta(x - x_0, y - y_0) = f(x_0, y_0)$$

2D FOURIER TRANSFORM: properties

Translation

$$f(x,y)e^{j2\pi(\mu_0 x/M+\nu_0 y/N)} \Leftrightarrow F(\mu-\mu_0,\nu-\nu_0)$$

and

$$f(x-x_0,y-y_0) \Leftrightarrow F(\mu,\nu)e^{-j2\pi(\mu x_0/M+\nu y_0/N)}$$

Convolution

$$f(x,y)*h(x,y) \Leftrightarrow F(u,v)H(u,v)$$

$$f(x,y)h(x,y) \Leftrightarrow F(u,v)*H(u,v)$$

2D FOURIER TRANSFORM: properties

Fourier Spectrum and Phase Angle

2-D DFT in polar form

$$F(u,v) = |F(u,v)|\, e^{j\phi(u,v)}$$

Fourier spectrum

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v) \right]^{1/2}$$

Power spectrum

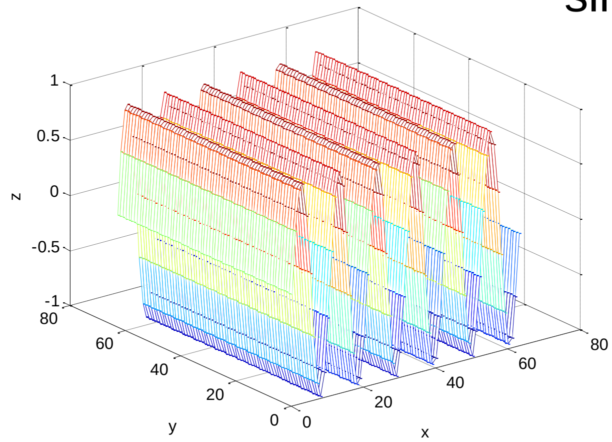
$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v)$$

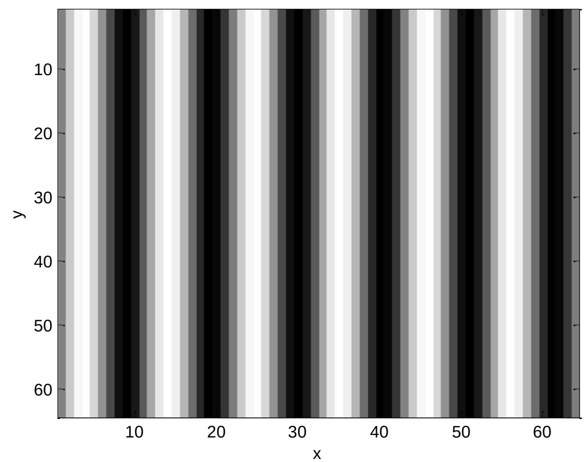
Phase angle

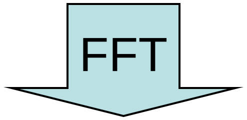
$$\phi(\mathrm{u,v}) = \mathrm{arctan}\left[\frac{I(u,v)}{R(u,v)} \right]$$

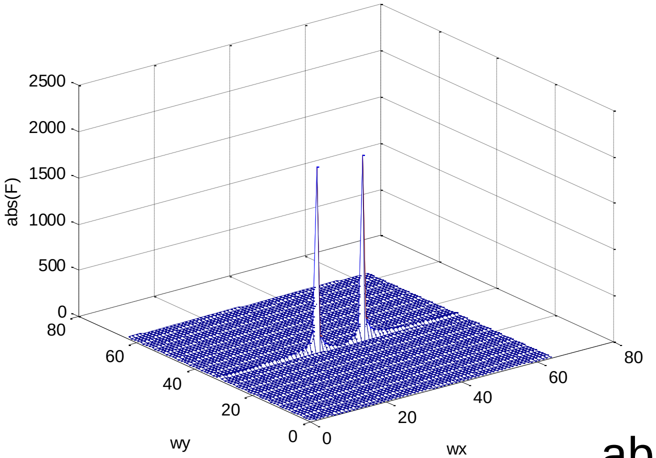
2D FOURIER TRANSFORM

sin() @ f1

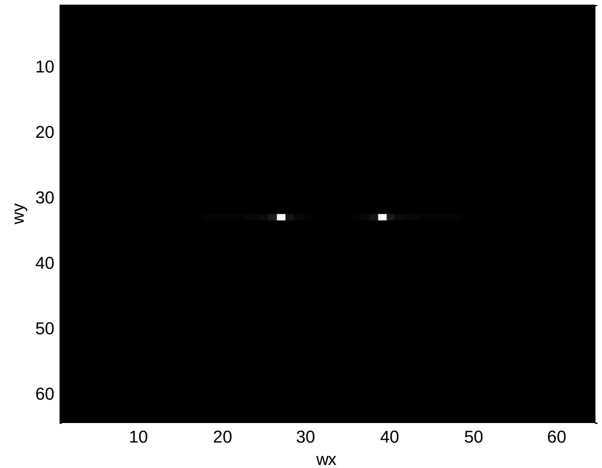








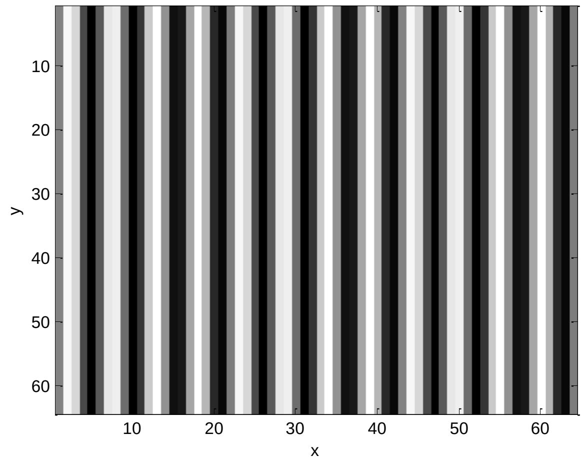
abs(FFT)



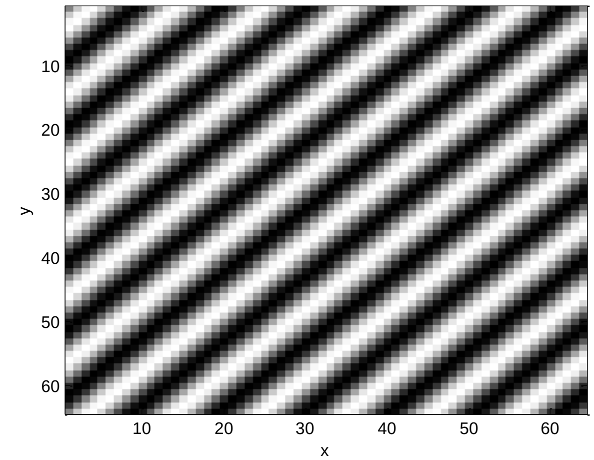
See ex_fft.m

2D FOURIER TRANSFORM

sin() @ f2>f1

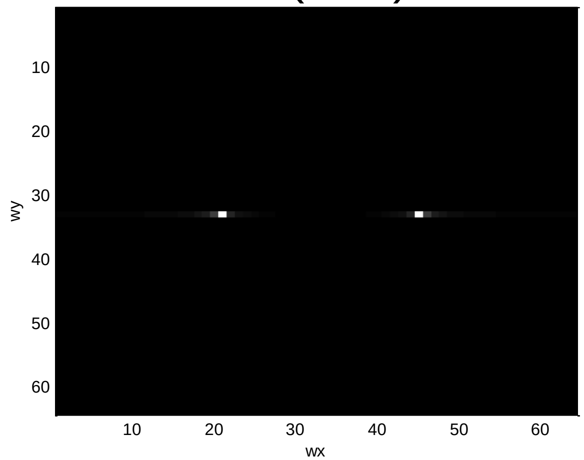


sin() @ f1, different orientation

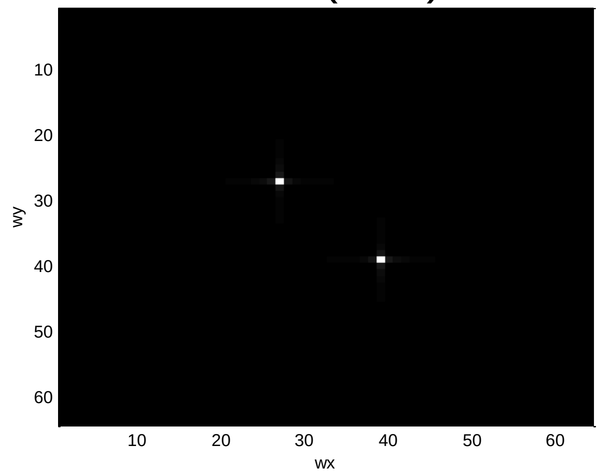


FFT

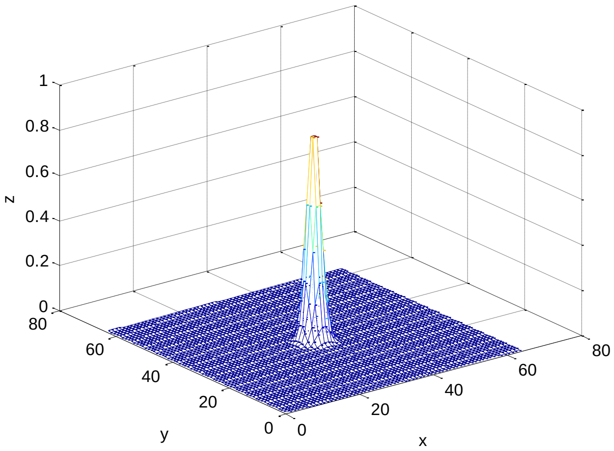
abs(FFT)



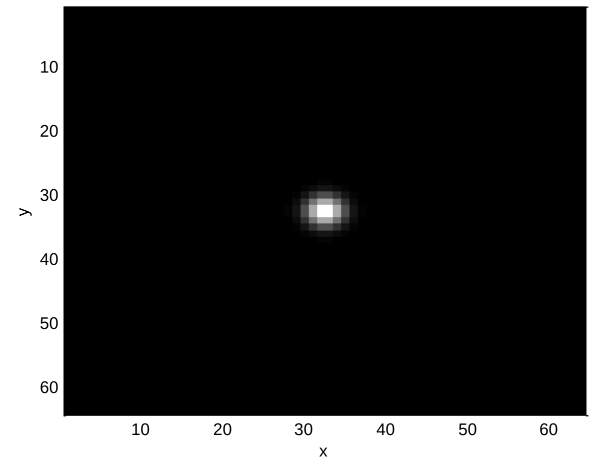
abs(FFT)



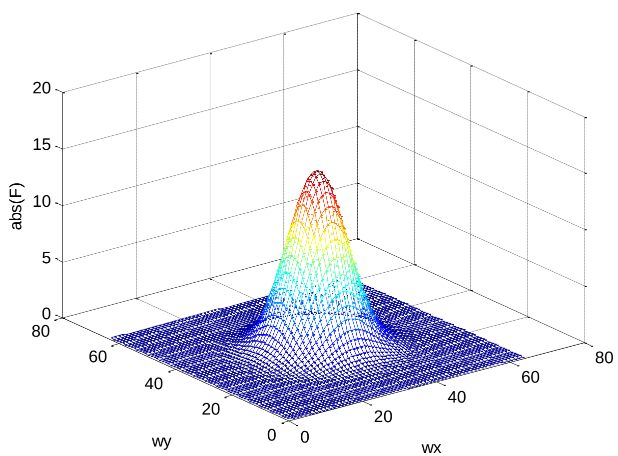
2D FOURIER TRANSFORM



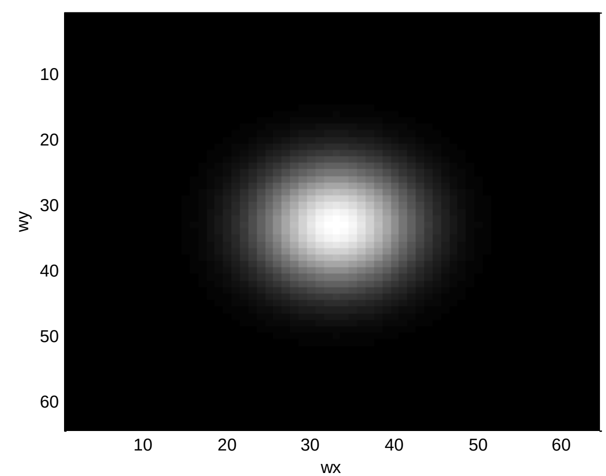
Gaussian



FFT

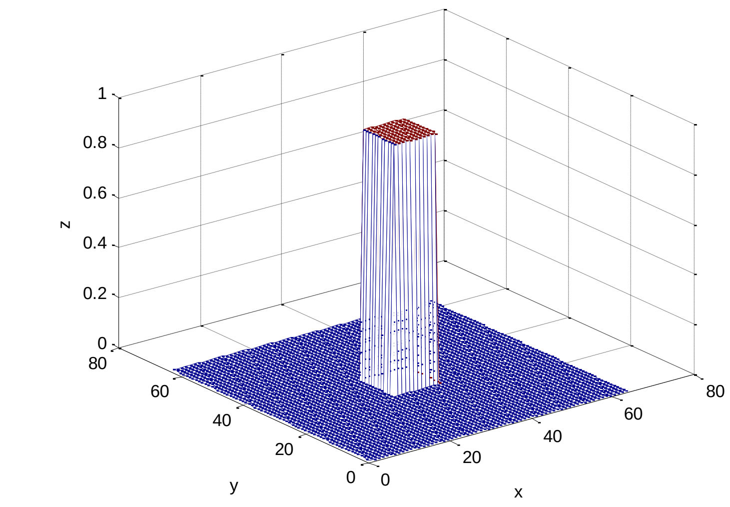


abs(FFT)

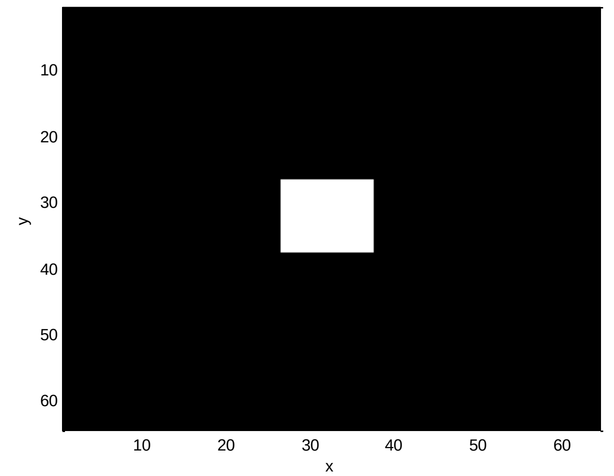


Low-pass filter

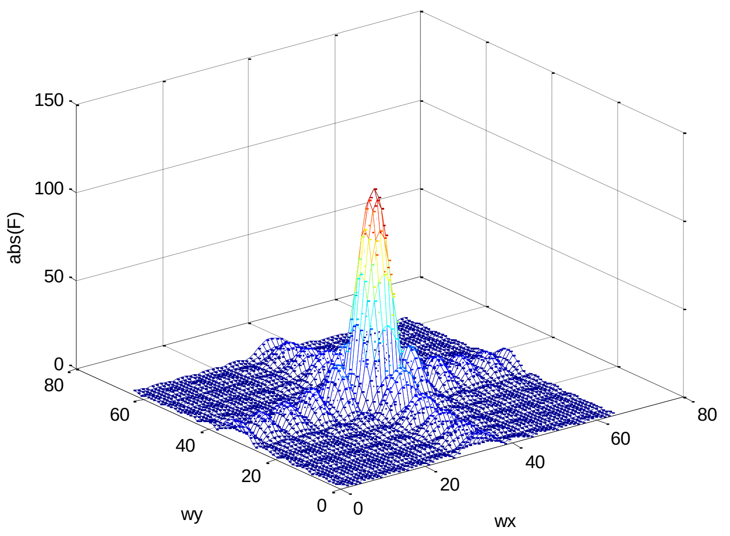
2D FOURIER TRANSFORM



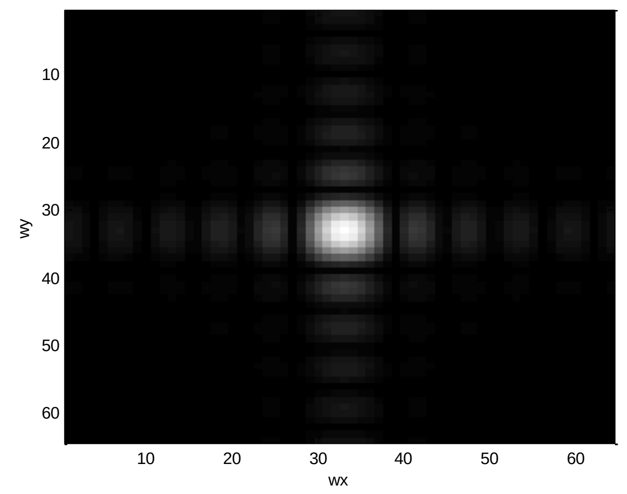
rectangle



FFT



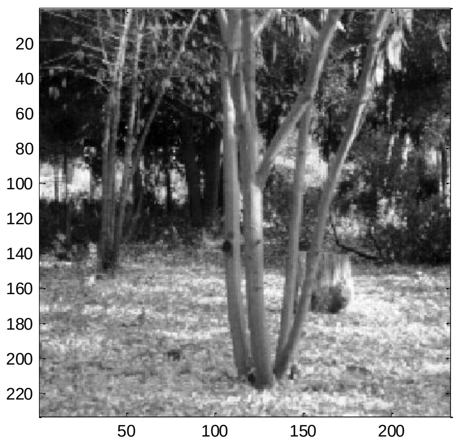
abs(FFT)



2D FOURIER TRANSFORM

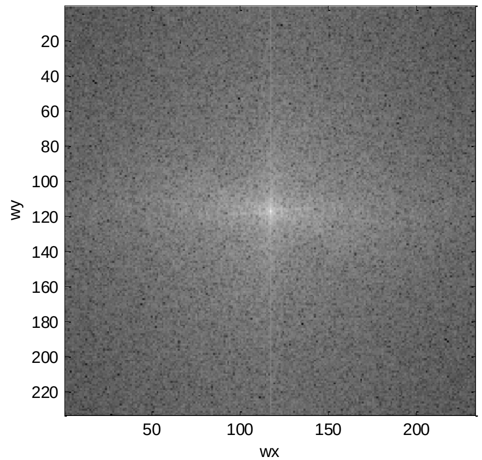
- The Fourier transform consists of a real and a complex component: the magnitude and phase, respectively.
- The value of the Fourier transform of a function/image at a particular (u,v) point depends on the whole function/image.
- The phase information is crucial to reconstruct the correct spatial structure of the image.

2D FOURIER TRANSFORM

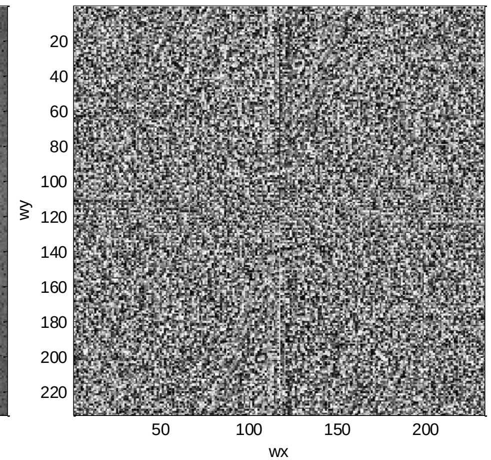


FFT

log(magnitude)

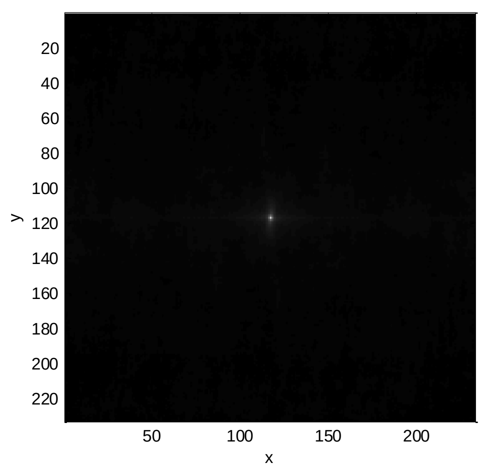


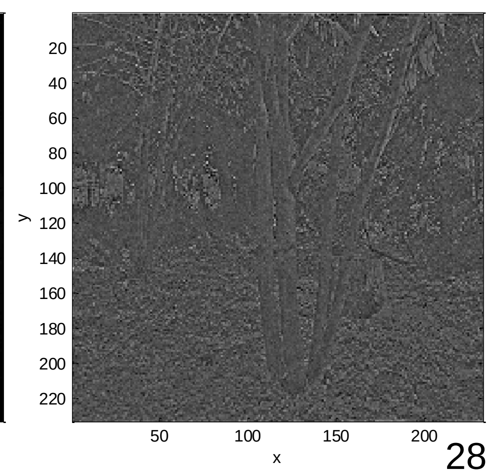
log(phase)



IFFT

IFFT





See phase.m

HOMEWORK

- Reading Assignments:
  - Mubarak Shah, “Fundamentals of Computer Vision”: sections 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.10.

- Lab : Friday, October 3, 11.00-13.00 (E5)