COM303: Digital Signal Processing

Lecture 20: Image Processing

overview

- Introduction to Images and Image Processing
- Affine Transforms
- 2D Fourier Analysis
- Image Filters

Overview:

- Images as multidimensional digital signals
- 2D signal representations
- Basic signals and operators

Overview:

- Images as multidimensional digital signals
- 2D signal representations
- Basic signals and operators

Overview:

- Images as multidimensional digital signals
- 2D signal representations
- Basic signals and operators

In the old, non-PC days...

Please meet ...

Digital images

- two-dimensional signal $x[n_1, n_2]$, $n_1, n_2 \in \mathbb{Z}$
- indices locate a point on a grid $\rightarrow$ pixel
- grid is usually regularly spaced
- values $x[n_1, n_2]$ refer to the pixel's appearance

Digital images

- two-dimensional signal $x[n_1, n_2]$, $n_1, n_2 \in \mathbb{Z}$
- indices locate a point on a grid $\rightarrow$ pixel
- grid is usually regularly spaced
- values $x[n_1, n_2]$ refer to the pixel's appearance

Digital images

- two-dimensional signal $x[n_1, n_2]$, $n_1, n_2 \in \mathbb{Z}$
- indices locate a point on a grid $\rightarrow$ pixel
- grid is usually regularly spaced
- values $x[n_1, n_2]$ refer to the pixel's appearance

Digital images

- two-dimensional signal $x[n_1, n_2]$, $n_1, n_2 \in \mathbb{Z}$
- indices locate a point on a grid $\rightarrow$ pixel
- grid is usually regularly spaced
- values $x[n_1, n_2]$ refer to the pixel's appearance

Digital images: grayscale vs color

- grayscale images: scalar pixel values
- color images: multidimensional pixel values in a color space (RGB, HSV, YUV, etc)
- we can consider the single components separately:

Digital images: grayscale vs color

- grayscale images: scalar pixel values
- color images: multidimensional pixel values in a color space (RGB, HSV, YUV, etc)
- we can consider the single components separately:

Digital images: grayscale vs color

- grayscale images: scalar pixel values
- color images: multidimensional pixel values in a color space (RGB, HSV, YUV, etc)
- we can consider the single components separately:

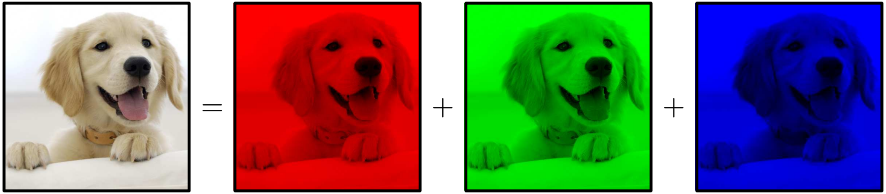


Image processing

From one to two dimensions...

- something still works
- something breaks down
- something is new

Image processing

From one to two dimensions...

- something still works
- something breaks down
- something is new

Image processing

From one to two dimensions...

- something still works
- something breaks down
- something is new

Image processing

What works:

- linearity, convolution
- Fourier transform
- interpolation, sampling

Image processing

What works:

- linearity, convolution
- Fourier transform
- interpolation, sampling

Image processing

What works:

- linearity, convolution
- Fourier transform
- interpolation, sampling

Image processing

What breaks down:

- Fourier analysis less relevant
- filter design hard, IIRs rare
- linear operators only mildly useful

Image processing

What breaks down:

- Fourier analysis less relevant
- filter design hard, IIRs rare
- linear operators only mildly useful

Image processing

What breaks down:

- Fourier analysis less relevant
- filter design hard, IIRs rare
- linear operators only mildly useful

Image processing

What's new:

- new manipulations: affine transforms
- images are finite-support signals
- images are (most often) available in their entirety $\rightarrow$ causality loses meaning
- images are very specialized signals, designed for a very specific processing system, i.e. the human brain! Lots of semantics that is extremely hard to deal with

Image processing

What's new:

- new manipulations: affine transforms
- images are finite-support signals
- images are (most often) available in their entirety $\rightarrow$ causality loses meaning
- images are very specialized signals, designed for a very specific processing system, i.e. the human brain! Lots of semantics that is extremely hard to deal with

Image processing

What's new:

- new manipulations: affine transforms
- images are finite-support signals
- images are (most often) available in their entirety $\rightarrow$ causality loses meaning
- images are very specialized signals, designed for a very specific processing system, i.e. the human brain! Lots of semantics that is extremely hard to deal with

Image processing

What's new:

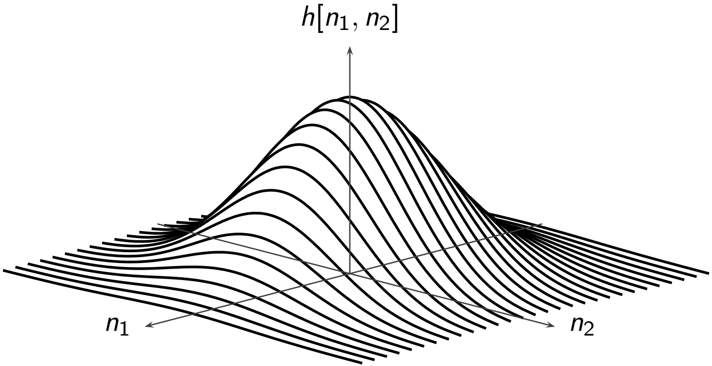
- new manipulations: affine transforms
- images are finite-support signals
- images are (most often) available in their entirety $\rightarrow$ causality loses meaning
- images are very specialized signals, designed for a very specific processing system, i.e. the human brain! Lots of semantics that is extremely hard to deal with

2D signal processing: the basics

A two-dimensional discrete-space signal:

$$x[n_1, n_2], \qquad n_1, n_2 \in \mathbb{Z}$$

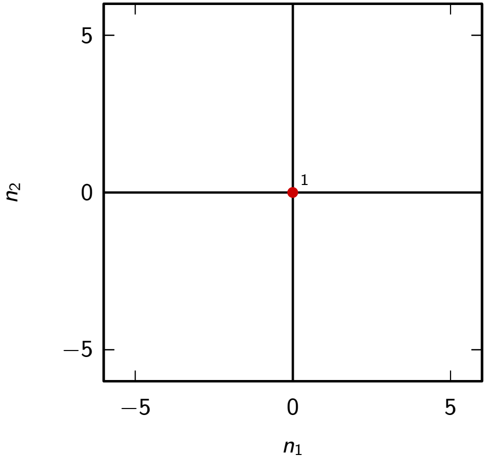
2D signals: Cartesian representation



2D signals: support representation

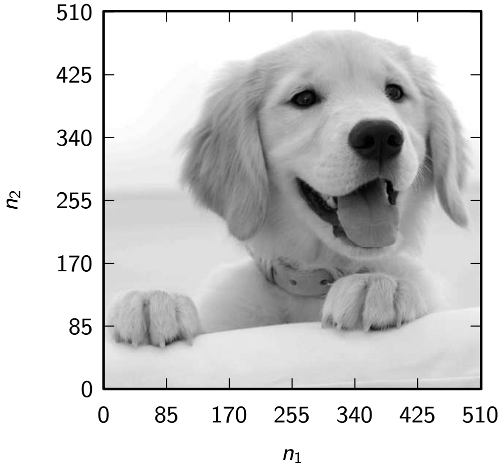
- just show coordinates of nonzero samples
- amplitude may be written along explicitly
- example:

$$\delta[n_1, n_2] = \begin{cases} 1 & \text{if } n_1 = n_2 = 0 \\ 0 & \text{otherwise.} \end{cases}$$



2D signals: image representation

- medium has a certain dynamic range (paper, screen)
- image values are quantized (usually to 8 bits, or 256 levels)
- the eye does the interpolation in space provided the pixel density is high enough



About dynamic ranges...

Images:

- human eye: 120dB
- prints: 12dB to 36dB
- LCD: 60dB
- digital cinema: 90dB

Sounds:

- human ear: 140dB
- speech: 40dB
- vinyl, tape: 50dB
- CD: 96dB

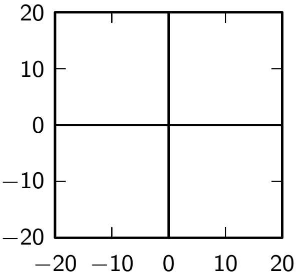
Why 2D?

- images could be unrolled (printers, fax)
- but what about spatial correlation?

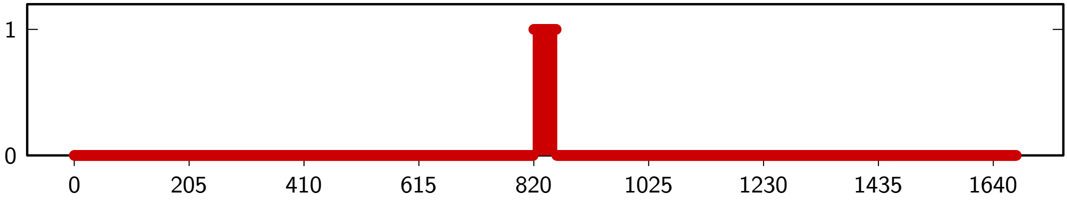
Why 2D?

- images could be unrolled (printers, fax)
- but what about spatial correlation?

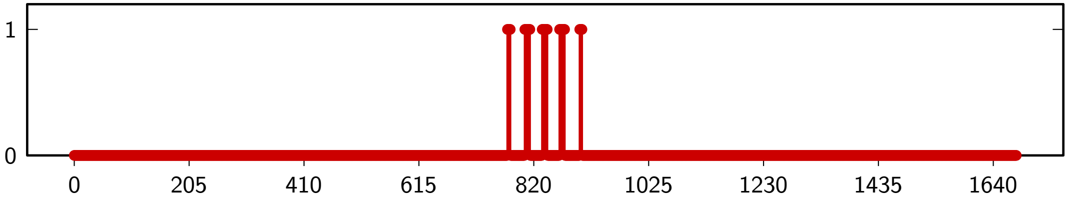
2D vs raster scan



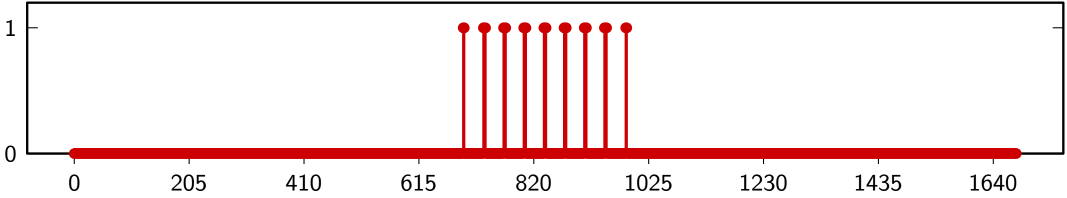
2D vs raster scan



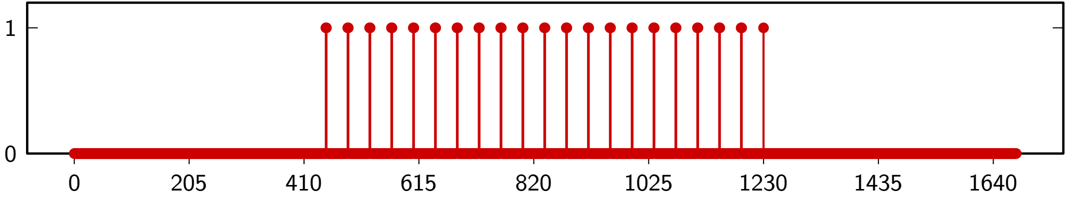
2D vs raster scan



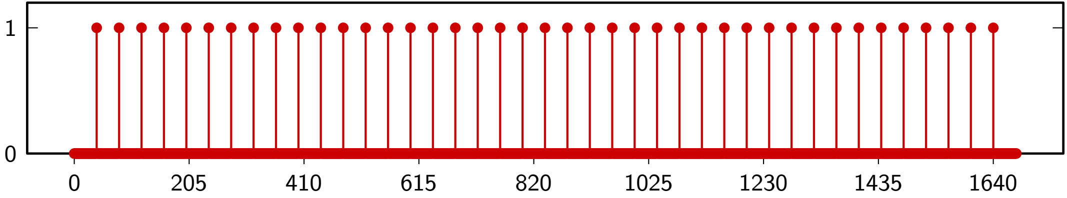
2D vs raster scan



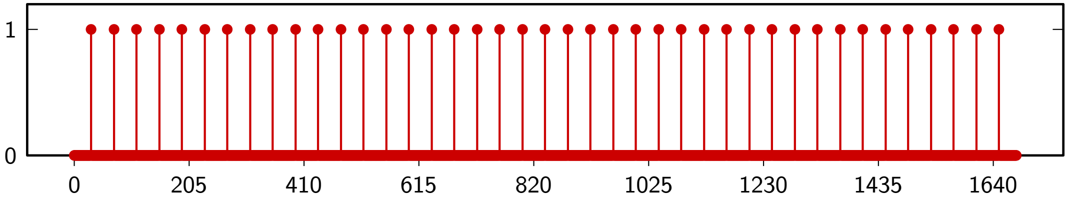
2D vs raster scan



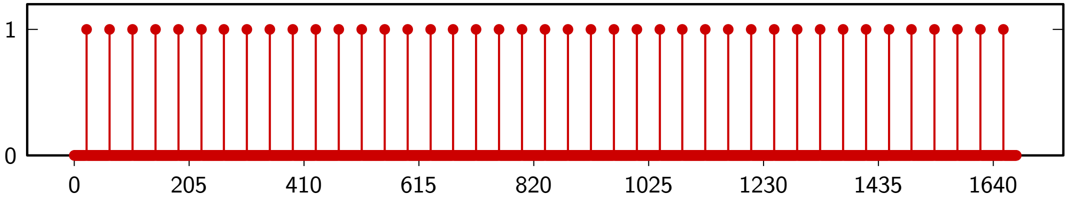
2D vs raster scan



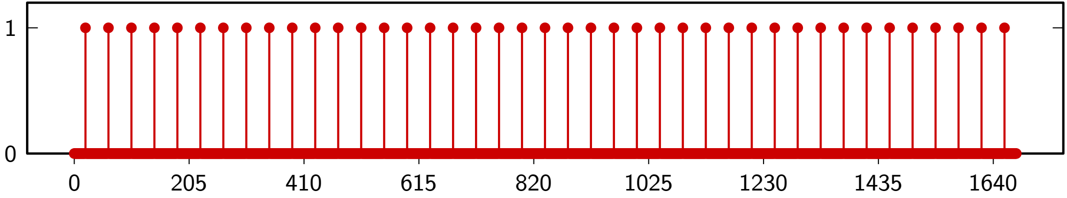
2D vs raster scan



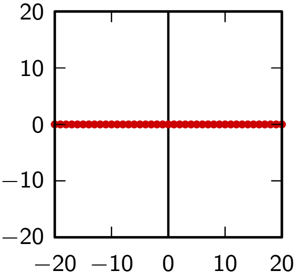
2D vs raster scan

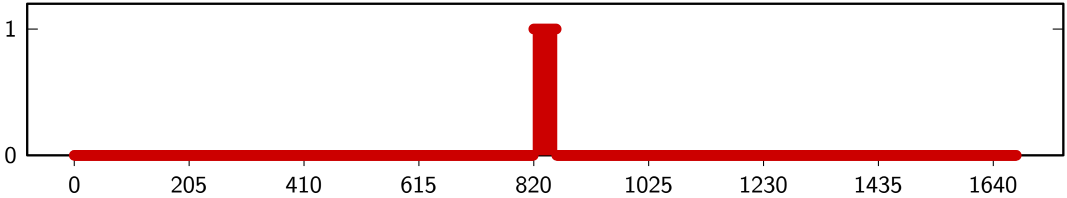


2D vs raster scan

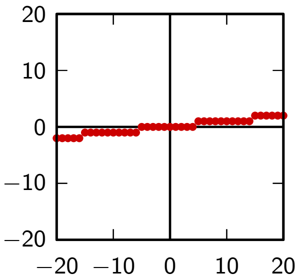


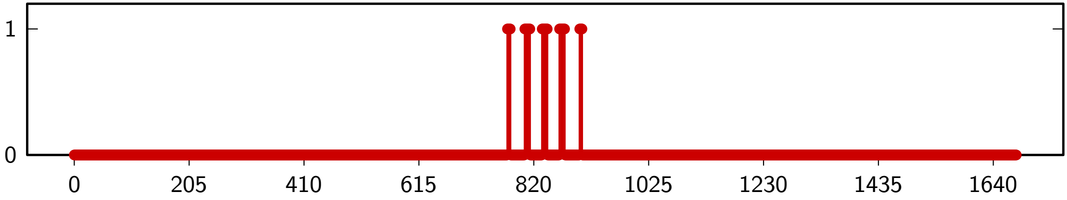
2D vs raster scan



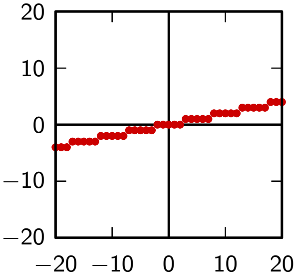


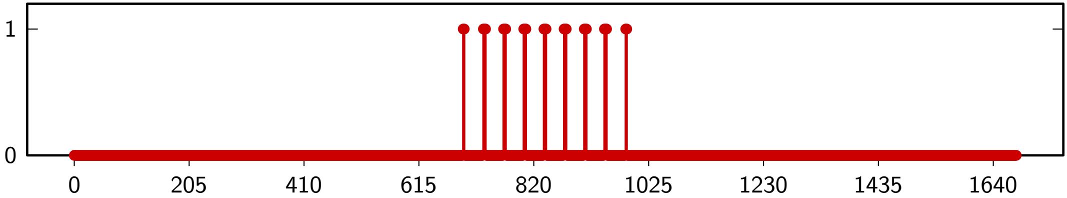
2D vs raster scan



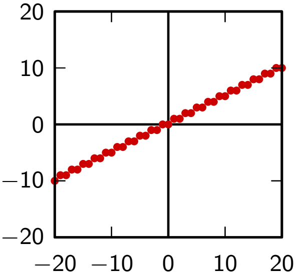


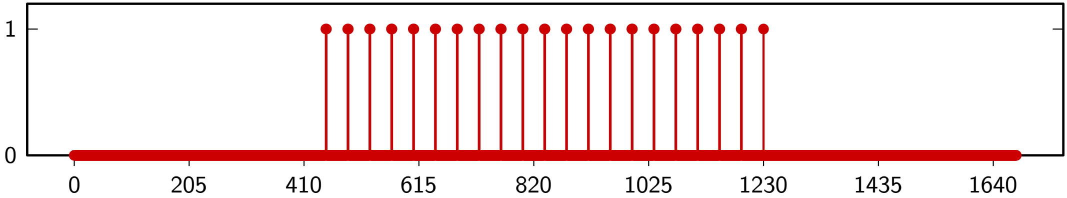
2D vs raster scan



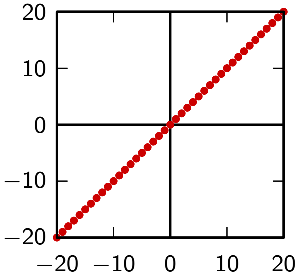


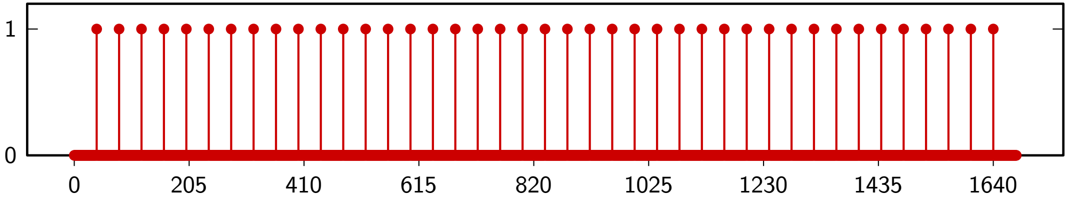
2D vs raster scan



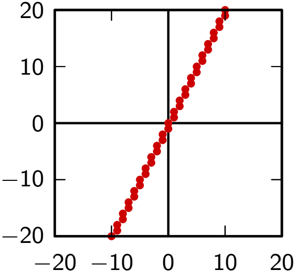


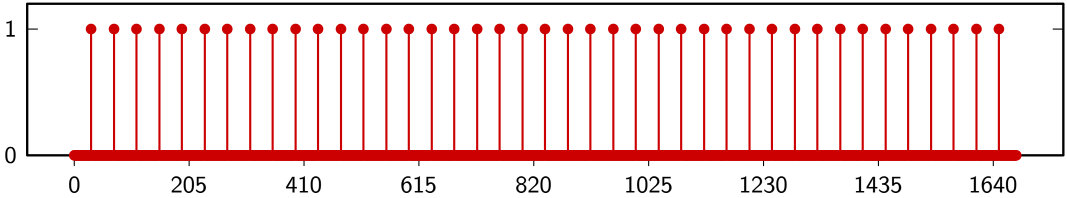
2D vs raster scan



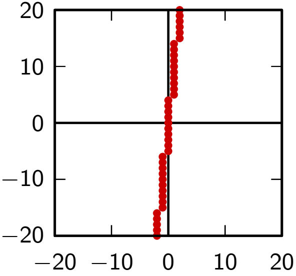


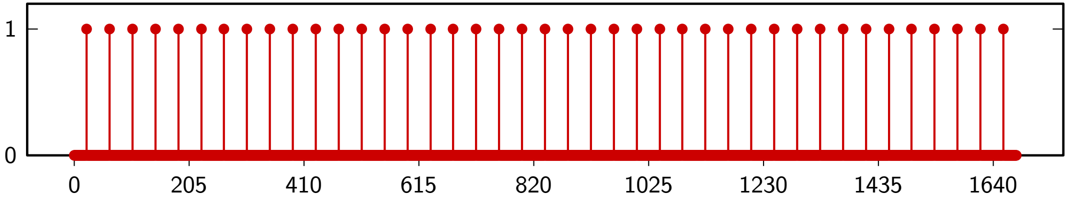
2D vs raster scan



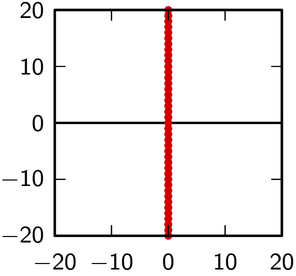


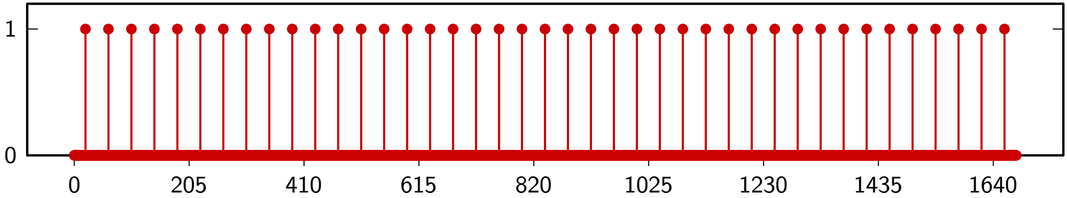
2D vs raster scan





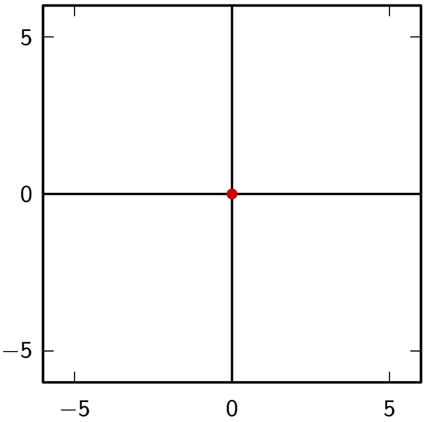
2D vs raster scan





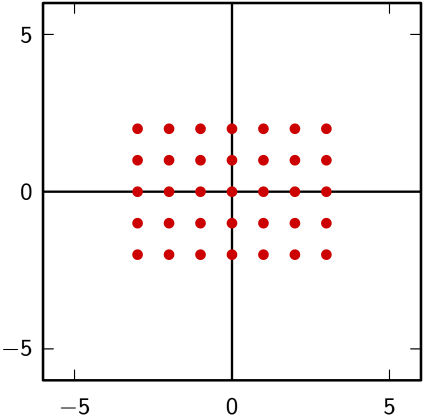
Basic 2D signals: the impulse

$$\delta[n_1, n_2] = \begin{cases} 1 & \text{if } n_1 = n_2 = 0 \\ 0 & \text{otherwise.} \end{cases}$$



Basic 2D signals: the rect

$$\mathrm{rect}\left(\frac{n_1}{2N_1}, \frac{n_2}{2N_2}\right) = \begin{cases} 1 & \text{if } |n_1| < N_1 \\ & \text{and } |n_2| < N_2 \\ 0 & \text{otherwise;} \end{cases}$$



Separability

$$x[n_1, n_2] = x_1[n_1]x_2[n_2]$$

Separable signals

$$\delta[n_1, n_2] = \delta[n_1]\delta[n_2]$$

$$\mathrm{rect}\left(\frac{n_1}{2N_1}, \frac{n_2}{2N_2}\right) = \mathrm{rect}\left(\frac{n_1}{2N_1}\right)\mathrm{rect}\left(\frac{n_2}{2N_2}\right).$$

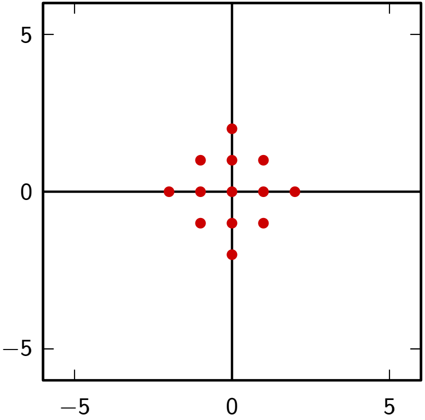
Separable signals

$$\delta[n_1, n_2] = \delta[n_1]\delta[n_2]$$

$$\operatorname{rect}\left(\frac{n_1}{2N_1}, \frac{n_2}{2N_2}\right) = \operatorname{rect}\left(\frac{n_1}{2N_1}\right)\operatorname{rect}\left(\frac{n_2}{2N_2}\right).$$

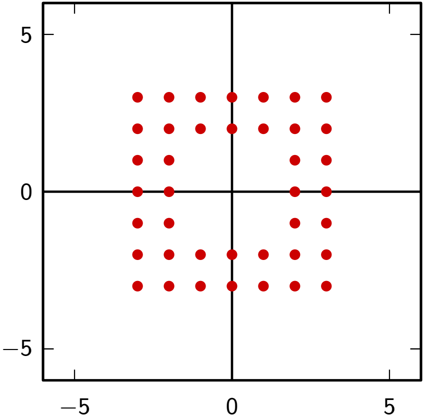
Nonseparable signal

$$x[n_1, n_2] = \begin{cases} 1 & \text{if } |n_1| + |n_2| < N \\ 0 & \text{otherwise} \end{cases}$$



Nonseparable signal

$$x[n_1, n_2] = \mathrm{rect}\left(\frac{n_1}{2N_1}, \frac{n_2}{2N_2}\right) - \mathrm{rect}\left(\frac{n_1}{2M_1}, \frac{n_2}{2M_2}\right)$$



2D convolution

$$x[n_1, n_2] * h[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x[k_1, k_2] h[n_1 - k_1, n_2 - k_2]$$

2D convolution for separable signals

If $h[n_1, n_2] = h_1[n_1]h_2[n_2]$:

$$x[n_1, n_2] * h[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} h_1[n_1 - k_1] \sum_{k_2=-\infty}^{\infty} x[k_1, k_2]h_2[n_2 - k_2]$$

$$= h_1[n_1] * (h_2[n_2] * x[n_1, n_2]).$$

2D convolution for separable signals

If $h[n_1, n_2]$ is an $M_1 \times M_2$ finite-support signal:

- non-separable convolution: $M_1 M_2$ operations per output sample
- separable convolution: $M_1 + M_2$ operations per output sample!

affine transforms

Affine transforms

mapping $\mathbb{R}^2 \to \mathbb{R}^2$ that reshapes the coordinate system:

$$\begin{bmatrix} t_1' \\ t_2' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} - \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

$$\begin{bmatrix} t_1' \\ t_2' \end{bmatrix} = \mathbf{A} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} - \mathbf{d}$$

Affine transforms

mapping $\mathbb{R}^2 \to \mathbb{R}^2$ that reshapes the coordinate system:

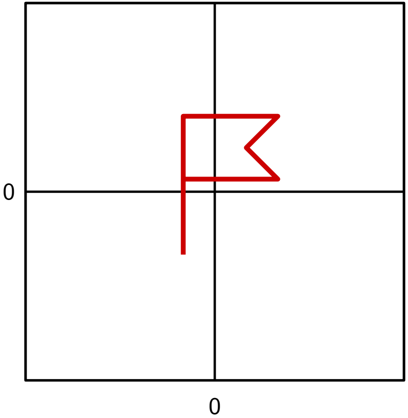
$$\begin{bmatrix} t_1' \\ t_2' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} - \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

$$\begin{bmatrix} t_1' \\ t_2' \end{bmatrix} = \mathbf{A} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} - \mathbf{d}$$

Translation

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}$$

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

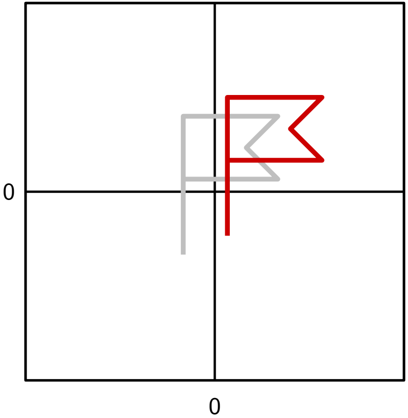


Translation

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}$$

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

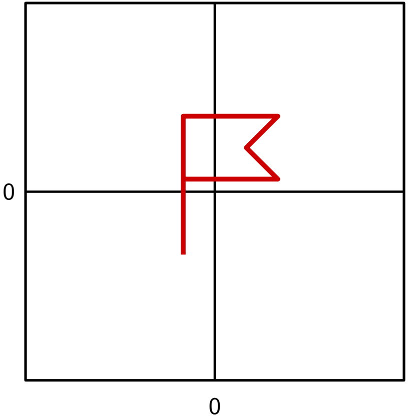
$d_1 = -0.7, d_2 = -0.3$



Scaling

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix}$$

$$\mathbf{d} = 0$$



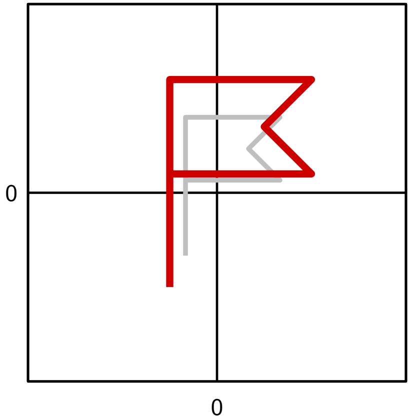
Scaling

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix}$$

$$\mathbf{d} = 0$$

if $a_1 = a_2$ the *aspect ratio* is preserved

$a_1 = a_2 = 1.5w$

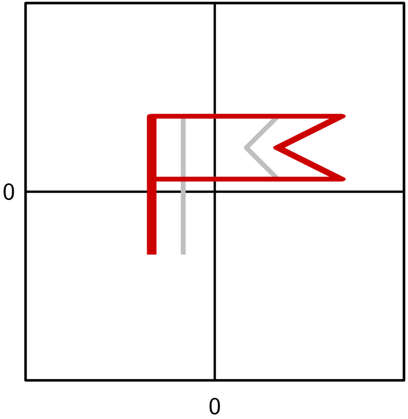


Scaling

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix}$$

$$\mathbf{d} = 0$$

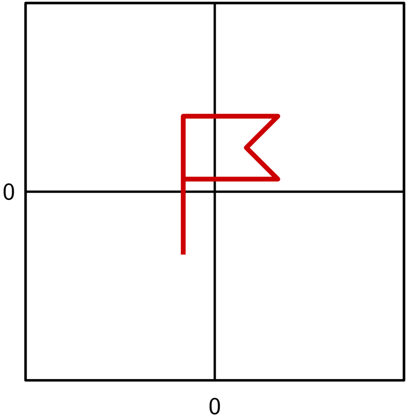
$a_1 = 2, a_2 = 1$



Rotation

$$\mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{d} = 0$$

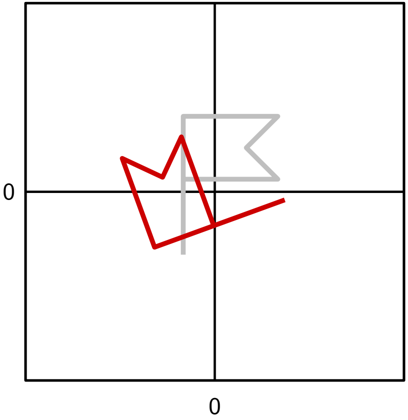


Rotation

$$\mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{d} = 0$$

$\theta = -0.6\pi$

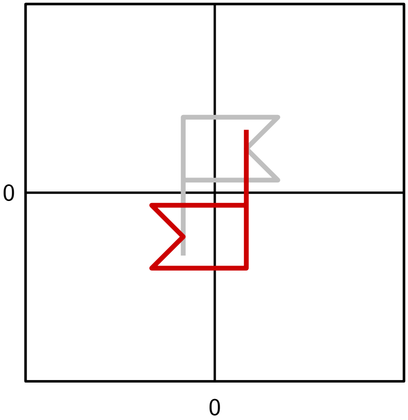


Rotation

$$\mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{d} = 0$$

$\theta = \pi$



Flips

Horizontal:

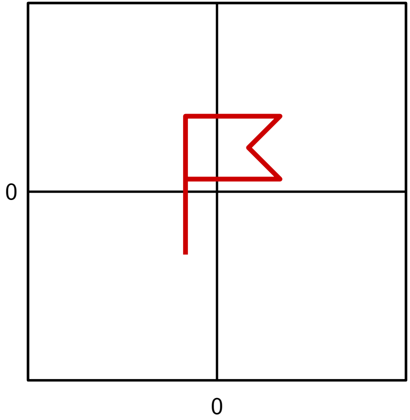
$$\mathbf{A} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$

Vertical:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\mathbf{d} = 0$$



Flips

Horizontal:

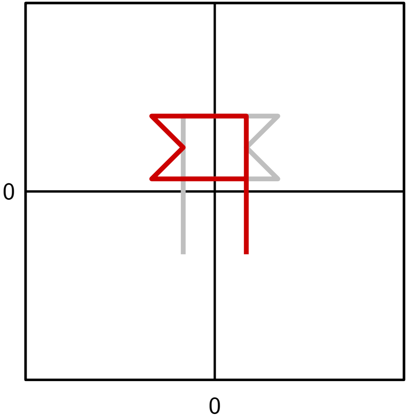
$$\mathbf{A} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$

Vertical:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\mathbf{d} = 0$$



Shear

Horizontal:

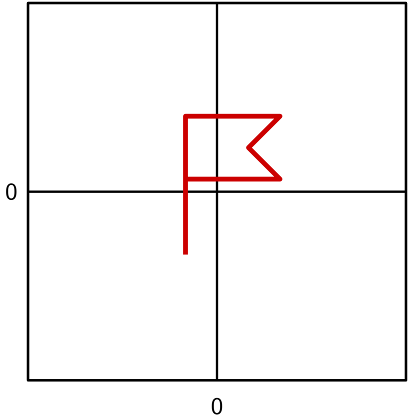
$$\mathbf{A} = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$

Vertical:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$



Shear

Horizontal:

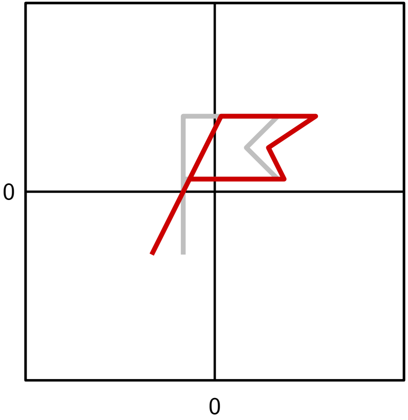
$$\mathbf{A} = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$

Vertical:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$



Affine transforms in discrete-space

$$\begin{bmatrix} t_1' \\ t_2' \end{bmatrix} = \mathbf{A} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} - \mathbf{d} \quad \in \mathbb{R}^2 \neq \mathbb{Z}^2$$

Solution for images

- take each *output point* $y[m_1, m_2]$
- apply the *inverse* transform to $[m_1, m_2]$ and find the *source* point's coordinates:

$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} m_1 + d_1 \\ m_2 + d_2 \end{bmatrix};$$

- if source point not on source grid, write

$$(t_1, t_2) = (\eta_1 + \tau_1, \eta_2 + \tau_2), \qquad \eta_{1,2} \in \mathbb{Z}, \quad 0 \leq \tau_{1,2} < 1$$

  and interpolate from the surrounding original grid points

Solution for images

- take each *output point* $y[m_1, m_2]$
- apply the *inverse* transform to $[m_1, m_2]$ and find the *source* point's coordinates:

$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} m_1 + d_1 \\ m_2 + d_2 \end{bmatrix};$$

- if source point not on source grid, write

$$(t_1, t_2) = (\eta_1 + \tau_1, \eta_2 + \tau_2), \qquad \eta_{1,2} \in \mathbb{Z}, \quad 0 \leq \tau_{1,2} < 1$$

and interpolate from the surrounding original grid points

Solution for images

- take each *output point* $y[m_1, m_2]$
- apply the *inverse* transform to $[m_1, m_2]$ and find the *source* point's coordinates:

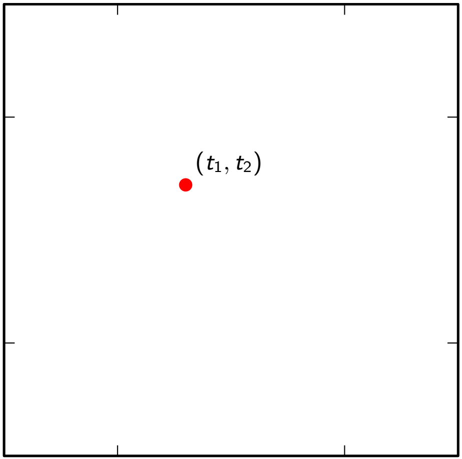
$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} m_1 + d_1 \\ m_2 + d_2 \end{bmatrix};$$

- if source point not on source grid, write

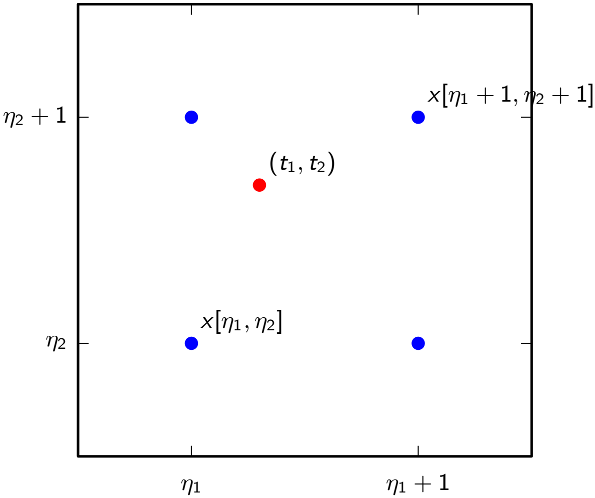
$$(t_1, t_2) = (\eta_1 + \tau_1, \eta_2 + \tau_2), \qquad \eta_{1,2} \in \mathbb{Z}, \quad 0 \leq \tau_{1,2} < 1$$

  and interpolate from the surrounding original grid points

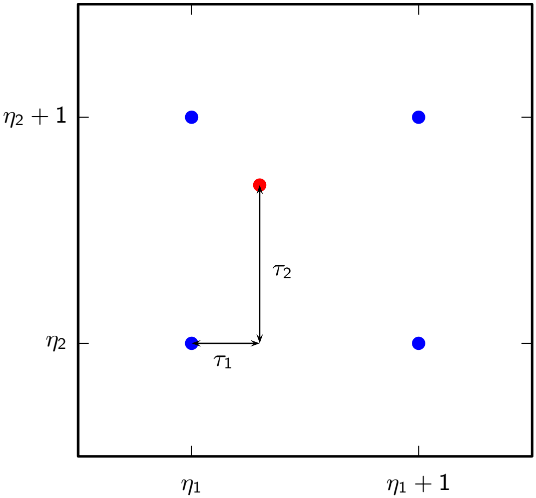
Bilinear Interpolation



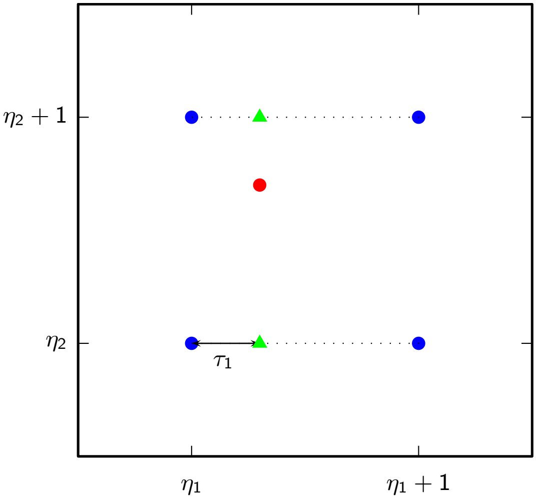
Bilinear Interpolation



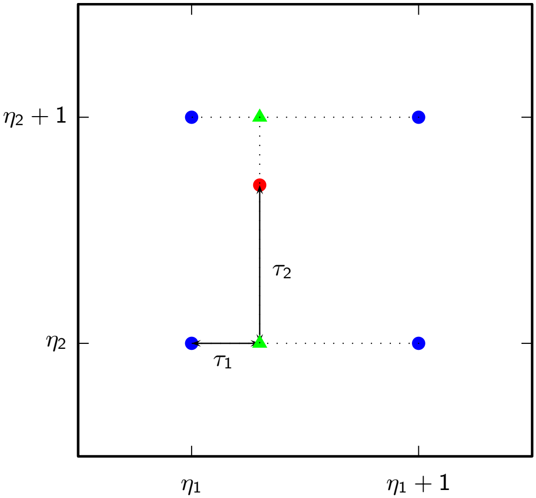
Bilinear Interpolation



Bilinear Interpolation



Bilinear Interpolation



Bilinear Interpolation

If we use a first-order interpolator:

$$\begin{aligned} y[m_1, m_2] = (1-\tau_1)(1-\tau_2)x[\eta_1, \eta_2] + \tau_1(1-\tau_2)x[\eta_1+1, \eta_2] \\ + (1-\tau_1)\tau_2 x[\eta_1, \eta_2+1] + \tau_1\tau_2 x[\eta_1+1, \eta_2+1] \end{aligned}$$

Shearing

2D Fourier Analysis

2D DFT

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_1} n_1 k_1} e^{-j\frac{2\pi}{N_2} n_2 k_2}$$

$$x[n_1, n_2] = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X[k_1, k_2] e^{j\frac{2\pi}{N_1} n_1 k_1} e^{j\frac{2\pi}{N_2} n_2 k_2}$$

2D DFT

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_1} n_1 k_1} e^{-j\frac{2\pi}{N_2} n_2 k_2}$$

$$x[n_1, n_2] = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X[k_1, k_2] e^{j\frac{2\pi}{N_1} n_1 k_1} e^{j\frac{2\pi}{N_2} n_2 k_2}$$

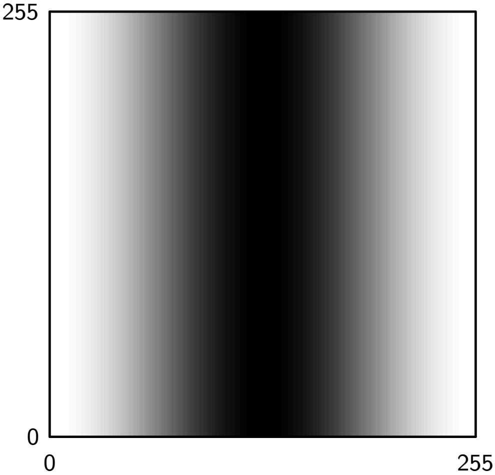
2D-DFT Basis Vectors

There are $N_1 N_2$ orthogonal basis vectors for an $N_1 \times N_2$ image:

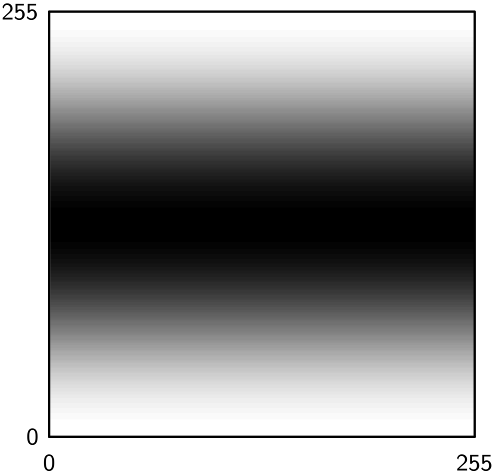
$$w_{k_1,k_2}[n_1, n_2] = e^{j\frac{2\pi}{N_1}n_1 k_1} e^{j\frac{2\pi}{N_2}n_2 k_2}$$

for $n_1, k_1 = 0, 1, \ldots, N_1 - 1$ and $n_2, k_2 = 0, 1, \ldots, N_2 - 1$

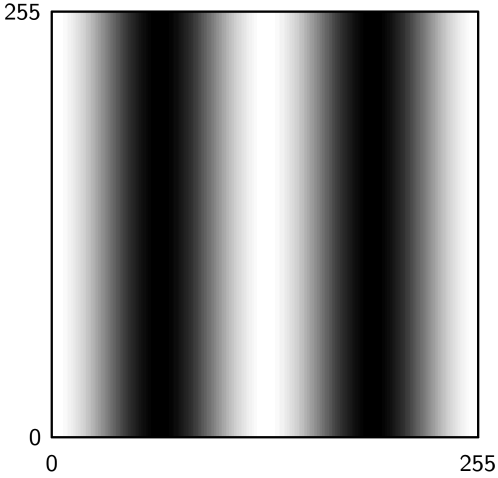
2D-DFT basis vectors for $k_1 = 1, k_2 = 0$ (real part)



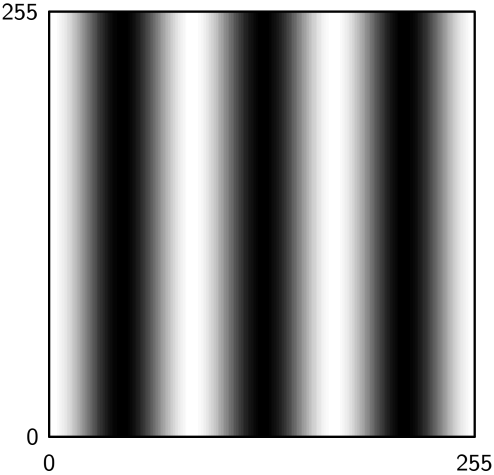
2D-DFT basis vectors for $k_1 = 0, k_2 = 1$ (real part)



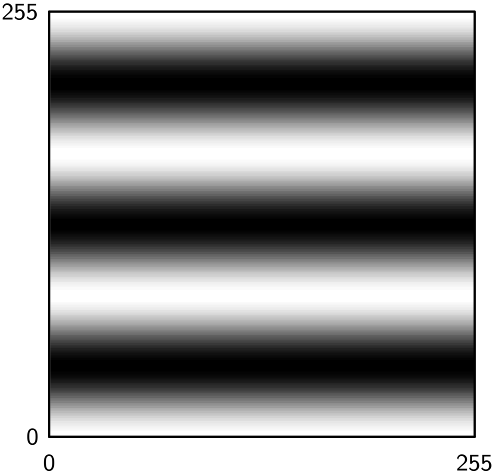
2D-DFT basis vectors for $k_1 = 2, k_2 = 0$ (real part)



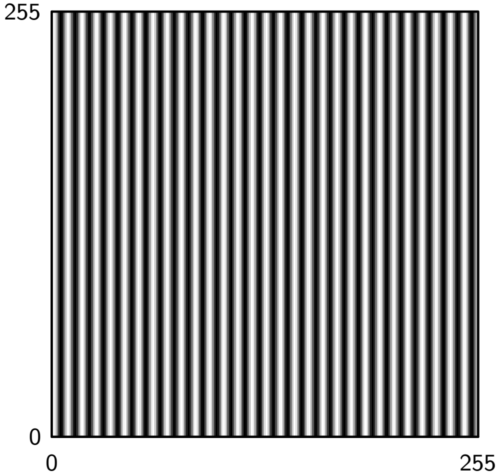
2D-DFT basis vectors for $k_1 = 3, k_2 = 0$ (real part)



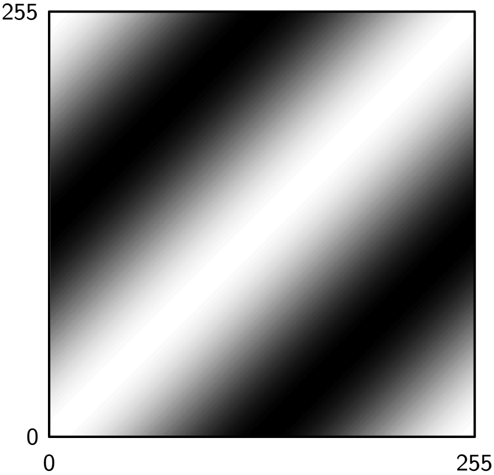
2D-DFT basis vectors for $k_1 = 0, k_2 = 3$ (real part)



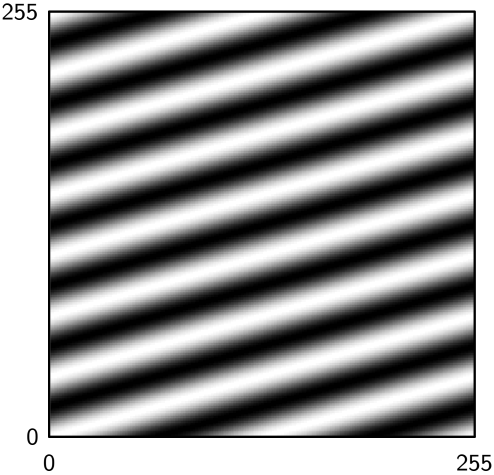
2D-DFT basis vectors for $k_1 = 30, k_2 = 0$ (real part)



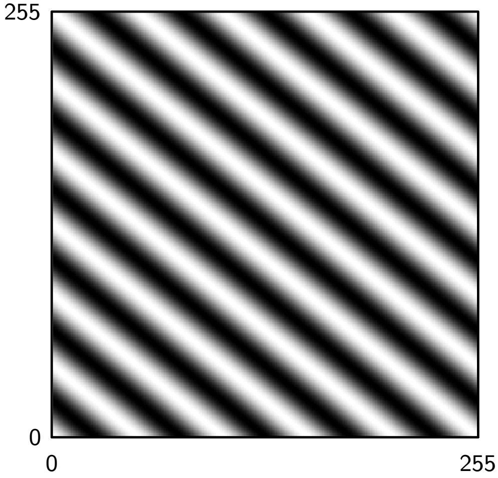
2D-DFT basis vectors for $k_1 = 1, k_2 = 1$ (real part)



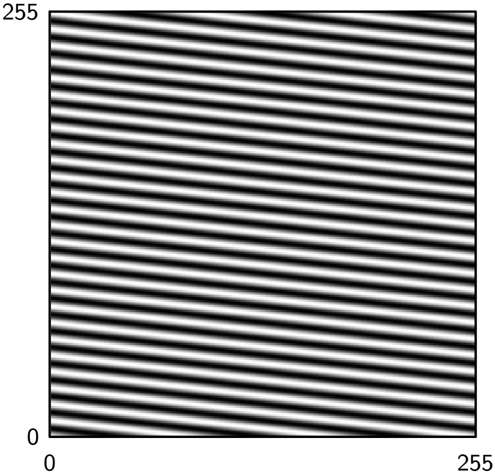
2D-DFT basis vectors for $k_1 = 2, k_2 = 7$ (real part)



2D-DFT basis vectors for $k_1 = 5, k_2 = 250$ (real part)



2D-DFT basis vectors for $k_1 = 3, k_2 = 230$ (real part)



2D DFT

2D-DFT basis functions are separable, and so is the 2D-DFT:

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_1} n_1 k_1} e^{-j\frac{2\pi}{N_2} n_2 k_2}$$

- 1D-DFT along n_2 (the columns)
- 1D-DFT along n_1 (the rows)

2D DFT

2D-DFT basis functions are separable, and so is the 2D-DFT:

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}$$

- 1D-DFT along n_2 (the columns)
- 1D-DFT along n_1 (the rows)

2D DFT

2D-DFT basis functions are separable, and so is the 2D-DFT:

$$X[k_1,k_2] = \sum_{n_1=0}^{N_1-1} \left[\boxed{\sum_{n_2=0}^{N_2-1} x[n_1,n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2}} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}$$

- 1D-DFT along n_2 (the columns)
- 1D-DFT along n_1 (the rows)

2D DFT

2D-DFT basis functions are separable, and so is the 2D-DFT:

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2}n_2 k_2} \right] e^{-j\frac{2\pi}{N_1}n_1 k_1}$$

- 1D-DFT along n_2 (the columns)
- 1D-DFT along n_1 (the rows)

2D DFT in matrix form

- finite-support 2D signal can be written as a matrix $\mathbf{x}$
- $N_1 \times N_2$ image is an $N_2 \times N_1$ matrix (n_1 spans the columns, n_2 spans the rows)
- recall also the $N \times N$ DFT matrix:

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^1 & W_N^2 & W^3 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W^4 & W_N^6 & \dots & W_N^{2(N-1)} \\ & & & \dots & & \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & W_N^{3(N-1)} & \dots & W_N^{(N-1)^2} \end{bmatrix}$$

2D DFT in matrix form

- finite-support 2D signal can be written as a matrix $\mathbf{x}$
- $N_1 \times N_2$ image is an $N_2 \times N_1$ matrix (n_1 spans the columns, n_2 spans the rows)
- recall also the $N \times N$ DFT matrix:

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^1 & W_N^2 & W^3 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W^4 & W_N^6 & \dots & W_N^{2(N-1)} \\ & & & \dots & & \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & W_N^{3(N-1)} & \dots & W_N^{(N-1)^2} \end{bmatrix}$$

2D DFT in matrix form

- finite-support 2D signal can be written as a matrix $\mathbf{x}$
- $N_1 \times N_2$ image is an $N_2 \times N_1$ matrix (n_1 spans the columns, n_2 spans the rows)
- recall also the $N \times N$ DFT matrix:

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^1 & W_N^2 & W^3 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W^4 & W_N^6 & \dots & W_N^{2(N-1)} \\ & & & \dots & & \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & W_N^{3(N-1)} & \dots & W_N^{(N-1)^2} \end{bmatrix}$$

2D DFT in matrix form

$$X[k_1, k_2] = \boxed{\sum_{n_1=0}^{N_1-1} \left[\boxed{\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2}} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}}$$

2D DFT in matrix form

$$X[k_1, k_2] = \boxed{\sum_{n_1=0}^{N_1-1} \left[\boxed{\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2}} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}}$$

$$\mathbf{V} = \mathbf{W}_{N_2}\, \mathbf{x}$$

$$\mathbf{V} \in \mathbb{C}^{N_2 \times N_1}$$

2D DFT in matrix form

$$X[k_1, k_2] = \boxed{\sum_{n_1=0}^{N_1-1} \left[\boxed{\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2}} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}}$$

$$\mathbf{V} = \mathbf{W}_{N_2}\, \mathbf{x}$$

$$\mathbf{V} \in \mathbb{C}^{N_2 \times N_1}$$

$$\mathbf{X} = \mathbf{V}\, \mathbf{W}_{N_1}$$

$$\mathbf{X} \in \mathbb{C}^{N_2 \times N_1}$$

2D DFT in matrix form

$$X[k_1, k_2] = \boxed{\sum_{n_1=0}^{N_1-1} \left[\boxed{\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2}} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}}$$

$$\mathbf{V} = \mathbf{W}_{N_2}\, \mathbf{x}$$

$$\mathbf{V} \in \mathbb{C}^{N_2 \times N_1}$$

$$\mathbf{X} = \mathbf{V}\, \mathbf{W}_{N_1}$$

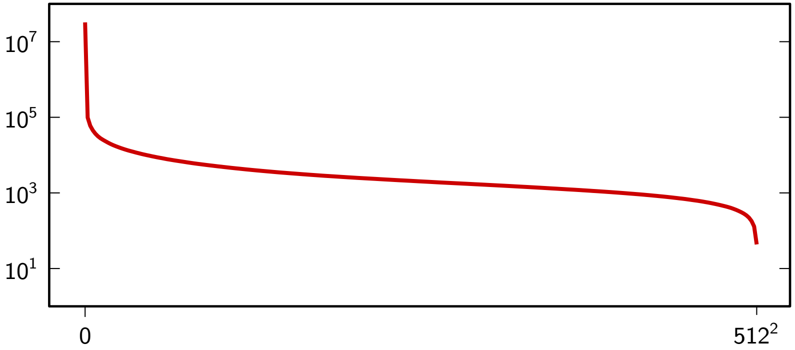
$$\mathbf{X} \in \mathbb{C}^{N_2 \times N_1}$$

$$\mathbf{X} = \mathbf{W}_{N_2}\, \mathbf{x}\, \mathbf{W}_{N_1}$$

How does a 2D-DFT look like?

- try to show the magnitude as an image
- problem: the range is too big for the grayscale range of paper or screen
- try to normalize: $|X'[n_1, n_2]| = |X[n_1, n_2]| / \max |X[n_1, n_2]|$
- but it doesn't work...

DFT coefficients sorted by magnitude



Dealing with HDR images

if the image is high dynamic range we need to compress the levels

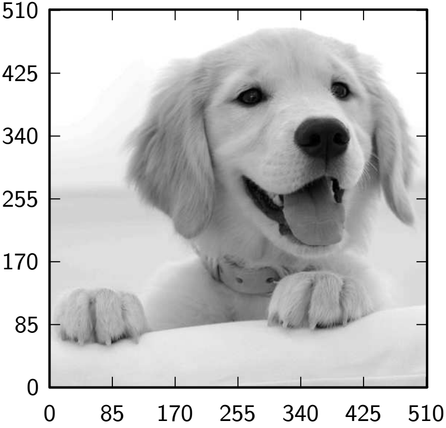
- remove flagrant outliers (e.g. $X[0,0] = \sum\sum x[n_1, n_2]$)
- use a nonlinear mapping: e.g. $y = x^{1/3}$ after normalization $(x \leq 1)$

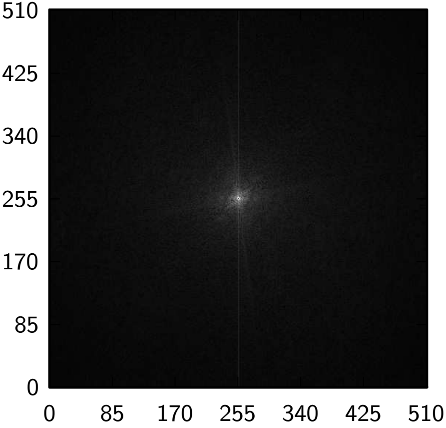
Dealing with HDR images

if the image is high dynamic range we need to compress the levels

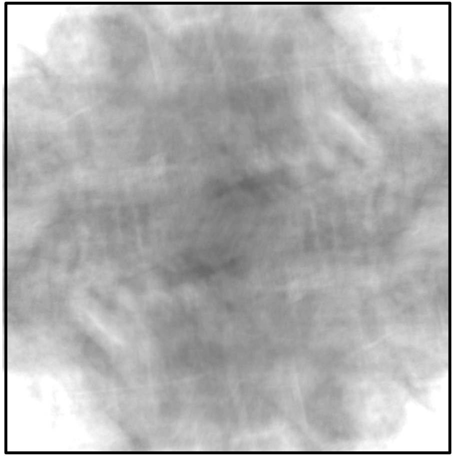
- remove flagrant outliers (e.g. $X[0,0] = \sum\sum x[n_1, n_2]$)
- use a nonlinear mapping: e.g. $y = x^{1/3}$ after normalization $(x \leq 1)$

How does a 2D-DFT look like?





DFT magnitude doesn't carry much information



DFT phase, on the other hand...

Image frequency analysis

- most of the information is contained in image's *edges*
- edges are points of abrupt change in signal's values
- edges are a "space-domain" feature $\rightarrow$ not captured by DFT's magnitude
- phase alignment is important for reproducing edges

image filtering

Overview:

- Filters for image processing
- Classification
- Examples

Overview:

- Filters for image processing
- Classification
- Examples

Overview:

- Filters for image processing
- Classification
- Examples

Analogies with 1D filters

- linearity
- *space* invariance
- impulse response
- frequency response
- stability
- 2D CCDE

The problem with LSI operators

- interesting images contain lots of *semantics*: different information in different areas
- space-invariant filters process everything in the same way
- but we should process things differently
  - edges
  - gradients
  - textures
  - ...

The problem with LSI operators

- interesting images contain lots of *semantics*: different information in different areas
- space-invariant filters process everything in the same way
- but we should process things differently
  - edges
  - gradients
  - textures
  - ...

The problem with LSI operators

- interesting images contain lots of *semantics*: different information in different areas
- space-invariant filters process everything in the same way
- but we should process things differently
  - edges
  - gradients
  - textures
  - ...

The problem with LSI operators

- interesting images contain lots of *semantics*: different information in different areas
- space-invariant filters process everything in the same way
- but we should process things differently
  - edges
  - gradients
  - textures
  - …

The problem with LSI operators

- interesting images contain lots of *semantics*: different information in different areas
- space-invariant filters process everything in the same way
- but we should process things differently
  - edges
  - gradients
  - textures
  - ...

The problem with LSI operators

- interesting images contain lots of *semantics*: different information in different areas
- space-invariant filters process everything in the same way
- but we should process things differently
  - edges
  - gradients
  - textures
  - ...

The problem with LSI operators

- interesting images contain lots of *semantics*: different information in different areas
- space-invariant filters process everything in the same way
- but we should process things differently
  - edges
  - gradients
  - textures
  - ...

Filter types

- IIR, FIR
- causal or noncausal
- highpass, lowpass, ...
  - lowpass $\rightarrow$ image smoothing
  - highpass $\rightarrow$ enhancement, edge detection

Filter types

- IIR, FIR
- causal or noncausal
- highpass, lowpass, ...
  - lowpass $\rightarrow$ image smoothing
  - highpass $\rightarrow$ enhancement, edge detection

Filter types

- IIR, FIR
- causal or noncausal
- highpass, lowpass, ...
  - lowpass $\rightarrow$ image smoothing
  - highpass $\rightarrow$ enhancement, edge detection

Filter types

- IIR, FIR
- causal or noncausal
- highpass, lowpass, ...
  - lowpass $\to$ image smoothing
  - highpass $\to$ enhancement, edge detection

Filter types

- IIR, FIR
- causal or noncausal
- highpass, lowpass, ...
  - lowpass $\rightarrow$ image smoothing
  - highpass $\rightarrow$ enhancement, edge detection

The problems with 2D IIRs

- nonlinear phase (edges!)
- border effects
- stability: the fundamental theorem of algebra doesn't hold in multiple dimensions!
- computability

The problems with 2D IIRs

- nonlinear phase (edges!)
- border effects
- stability: the fundamental theorem of algebra doesn't hold in multiple dimensions!
- computability

The problems with 2D IIRs

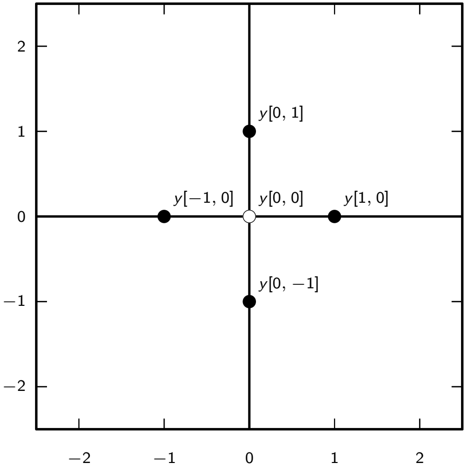
- nonlinear phase (edges!)
- border effects
- stability: the fundamental theorem of algebra doesn't hold in multiple dimensions!
- computability

The problems with 2D IIRs

- nonlinear phase (edges!)
- border effects
- stability: the fundamental theorem of algebra doesn't hold in multiple dimensions!
- computability

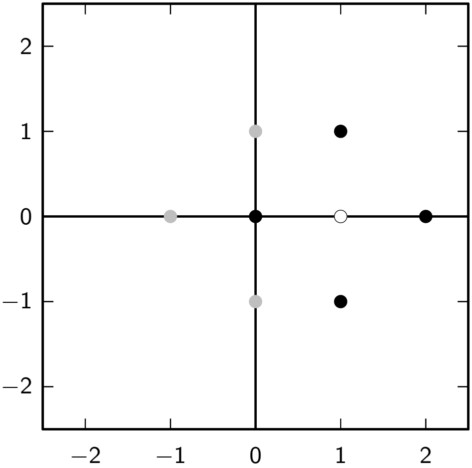
A noncomputable CCDE

$$y[n_1, n_2] = a_0 y[n_1 + 1, n_2] + a_1 y[n_1, n_2 - 1] + a_2 y[n_1 - 1, n_2] + a_3 y[n_1, n_2 + 1] + x[n_1, n_2];$$



A noncomputable CCDE

$$y[n_1, n_2] = a_0 y[n_1 + 1, n_2] + a_1 y[n_1, n_2 - 1] + a_2 y[n_1 - 1, n_2] + a_3 y[n_1, n_2 + 1] + x[n_1, n_2];$$



Practical FIR filters

- generally zero centered (causality not an issue) $\Rightarrow$ odd number of taps in both directions
- per-sample complexity:
  - $M_1 M_2$ for nonseparable impulse responses
  - $M_1 + M_2$ for separable impulse responses
- obviously always stable

Practical FIR filters

- generally zero centered (causality not an issue) $\Rightarrow$ odd number of taps in both directions
- per-sample complexity:
  - $M_1 M_2$ for nonseparable impulse responses
  - $M_1 + M_2$ for separable impulse responses
- obviously always stable

Practical FIR filters

- generally zero centered (causality not an issue) $\Rightarrow$ odd number of taps in both directions
- per-sample complexity:
  - $M_1 M_2$ for nonseparable impulse responses
  - $M_1 + M_2$ for separable impulse responses
- obviously always stable

Practical FIR filters

- generally zero centered (causality not an issue) $\Rightarrow$ odd number of taps in both directions
- per-sample complexity:
  - $M_1 M_2$ for nonseparable impulse responses
  - $M_1 + M_2$ for separable impulse responses
- obviously always stable

Practical FIR filters

- generally zero centered (causality not an issue) $\Rightarrow$ odd number of taps in both directions
- per-sample complexity:
  - $M_1 M_2$ for nonseparable impulse responses
  - $M_1 + M_2$ for separable impulse responses
- obviously always stable

Moving Average

$$y[n_1, n_2] = \frac{1}{(2N+1)^2} \sum_{k_1=-N}^{N} \sum_{k_2=-N}^{N} x[n_1 - k_1, n_2 - k_2]$$

$$h[n_1, n_2] = \frac{1}{(2N+1)^2} \mathrm{rect}\left(\frac{n_1}{2N}, \frac{n_2}{2N}\right)$$

Moving Average

$$y[n_1, n_2] = \frac{1}{(2N+1)^2} \sum_{k_1=-N}^{N} \sum_{k_2=-N}^{N} x[n_1 - k_1, n_2 - k_2]$$

$$h[n_1, n_2] = \frac{1}{(2N+1)^2} \mathrm{rect}\left(\frac{n_1}{2N}, \frac{n_2}{2N}\right)$$

Moving Average

$$h[n_1, n_2] = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Moving Average

11 × 11 MA

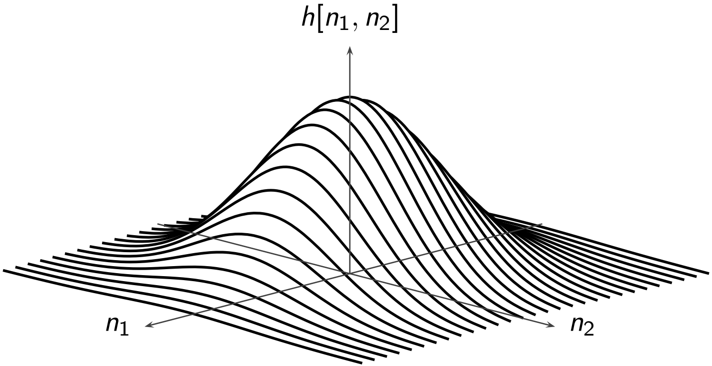
51 × 51 MA

Gaussian Blur

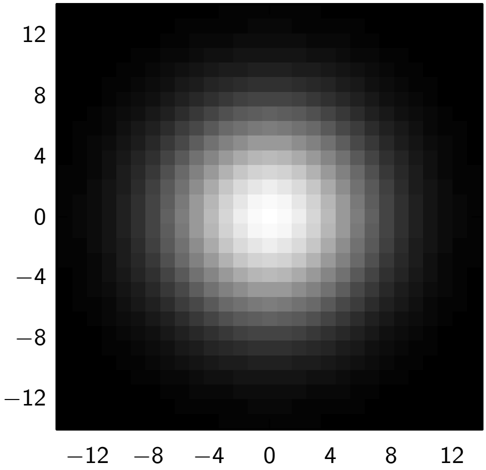
$$h[n_1, n_2] = \frac{1}{2\pi\sigma^2} e^{-\frac{n_1^2+n_2^2}{2\sigma^2}}, \qquad |n_1, n_2| < N$$

with $N \approx 3\sigma$

Gaussian Blur



Gaussian Blur



$\sigma = 5, N = 14$

Gaussian Blur

$\sigma = 1.8, 11 \times 11$ blur

$\sigma = 8.7, 51 \times 51$ blur

Sobel filter

approximation of the first derivative in the horizontal direction:

$$s_o[n_1, n_2] = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

separability and structure:

$$s_o[n_1, n_2] = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Sobel filter

approximation of the first derivative in the horizontal direction:

$$s_o[n_1, n_2] = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

separability and structure:

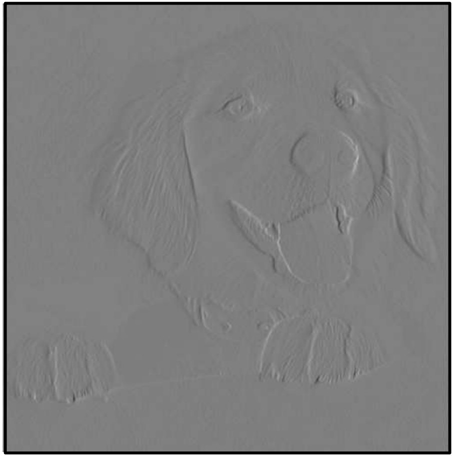
$$s_o[n_1, n_2] = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Sobel filter

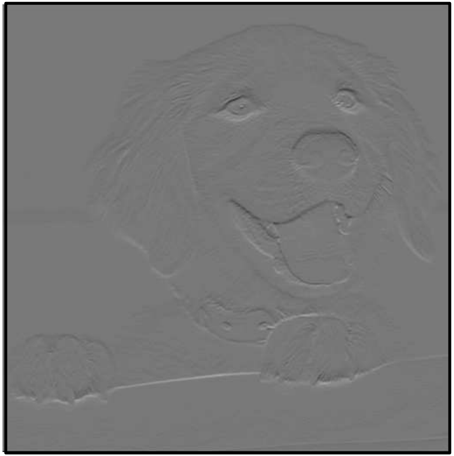
approximation of the first derivative in the vertical direction:

$$s_v[n_1, n_2] = \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel filter



horizontal Sobel filter



vertical Sobel filter

Sobel operator

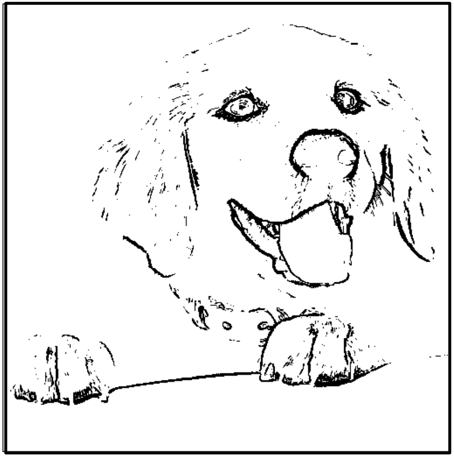
approximation for the square magnitude of the gradient:

$$|\nabla x[n_1, n_2]|^2 = |s_o[n_1, n_2] * x[n_1, n_2]|^2 + |s_v[n_1, n_2] * x[n_1, n_2]|^2$$

("operator" because it's nonlinear)

Gradient approximation for edge detection

Sobel operator



thresholeded Sobel operator

Laplacian operator

Laplacian of a function in continuous-space:

$$\Delta f(t_1, t_2) = \frac{\partial^2 f}{\partial t_1^2} + \frac{\partial^2 f}{\partial t_2^2}$$

Laplacian operator

approximating the Laplacian; start with a Taylor expansion

$$f(t+\tau) = \sum_{n=0}^{\infty} \frac{f^{(n)}(t)}{n!}\tau^n$$

and compute the expansion in $(t+\tau)$ and $(t-\tau)$:

$$f(t+\tau) = f(t) + f'(t)\tau + \frac{1}{2}f''(t)\tau^2$$

$$f(t-\tau) = f(t) - f'(t)\tau + \frac{1}{2}f''(t)\tau^2$$

Laplacian operator

by rearranging terms:

$$f''(t) = \frac{1}{\tau^2}(f(t-\tau) - 2f(t) + f(t+\tau))$$

which, on the discrete grid, is the FIR $h[n] = \begin{bmatrix}1 & -2 & 1\end{bmatrix}$

Laplacian

summing the horizontal and vertical components:

$$h[n_1, n_2] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

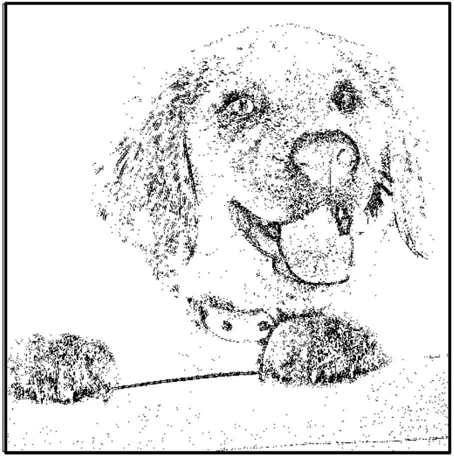
Laplacian

If we use the diagonals too:

$$h[n_1, n_2] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Laplacian for Edge Detection

Laplacian operator



thresholeded Laplacian operator