

lecture 2: representation

deep learning for vision

Yannis Avrithis

Inria Rennes-Bretagne Atlantique

Rennes, Nov. 2017 – Jan. 2018



outline

introduction

receptive fields

visual descriptors

embeddings

introduction

image retrieval challenges



image retrieval challenges



- scale
 - viewpoint
 - occlusion
 - clutter
 - lighting
-
- distinctiveness
 - distractors

image classification challenges



image classification challenges

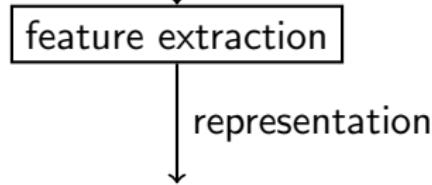


- scale
- viewpoint
- occlusion
- clutter
- lighting
- number of instances
- texture/color
- pose
- deformability
- intra-class variability

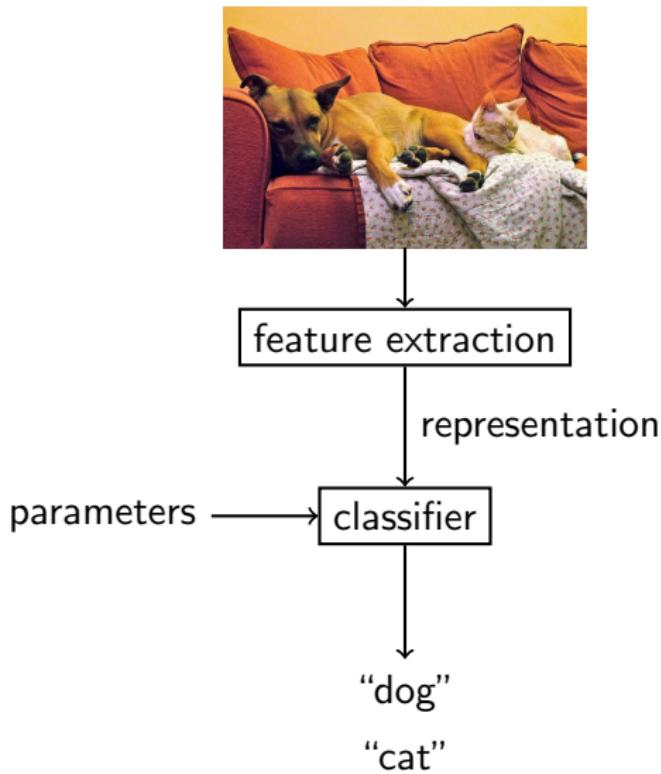
data-driven approach



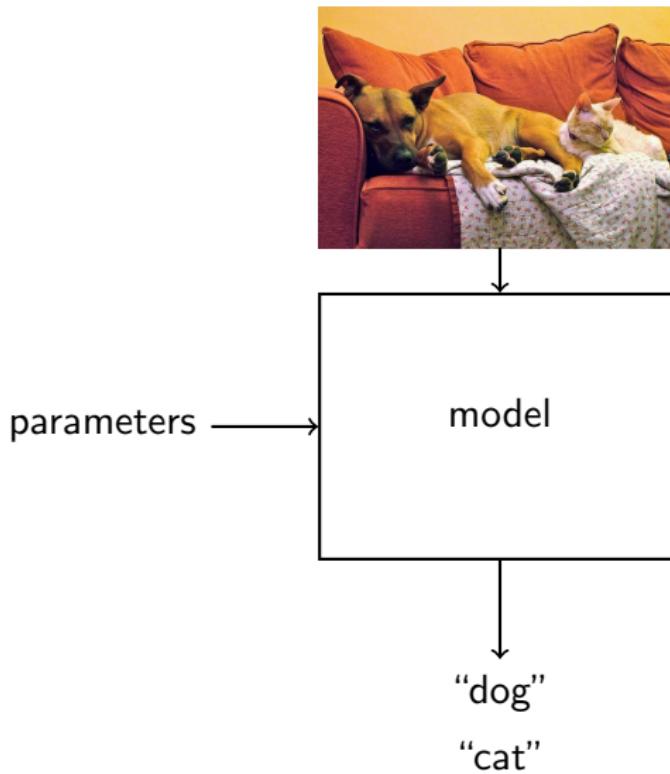
data-driven approach



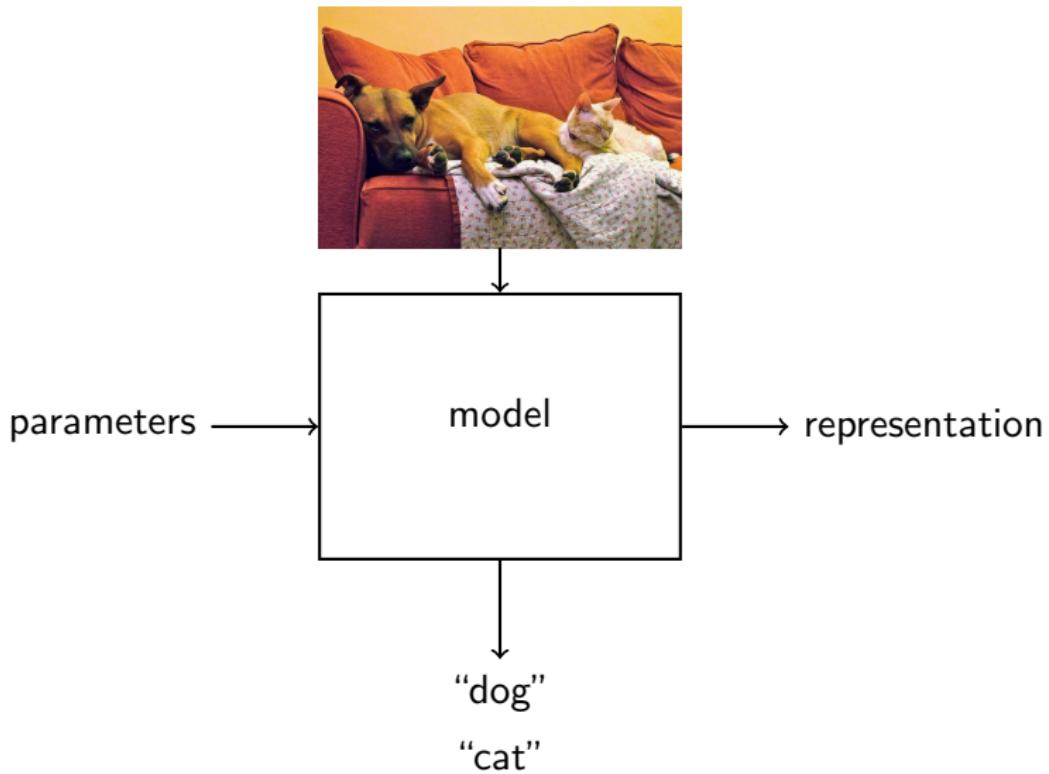
data-driven approach



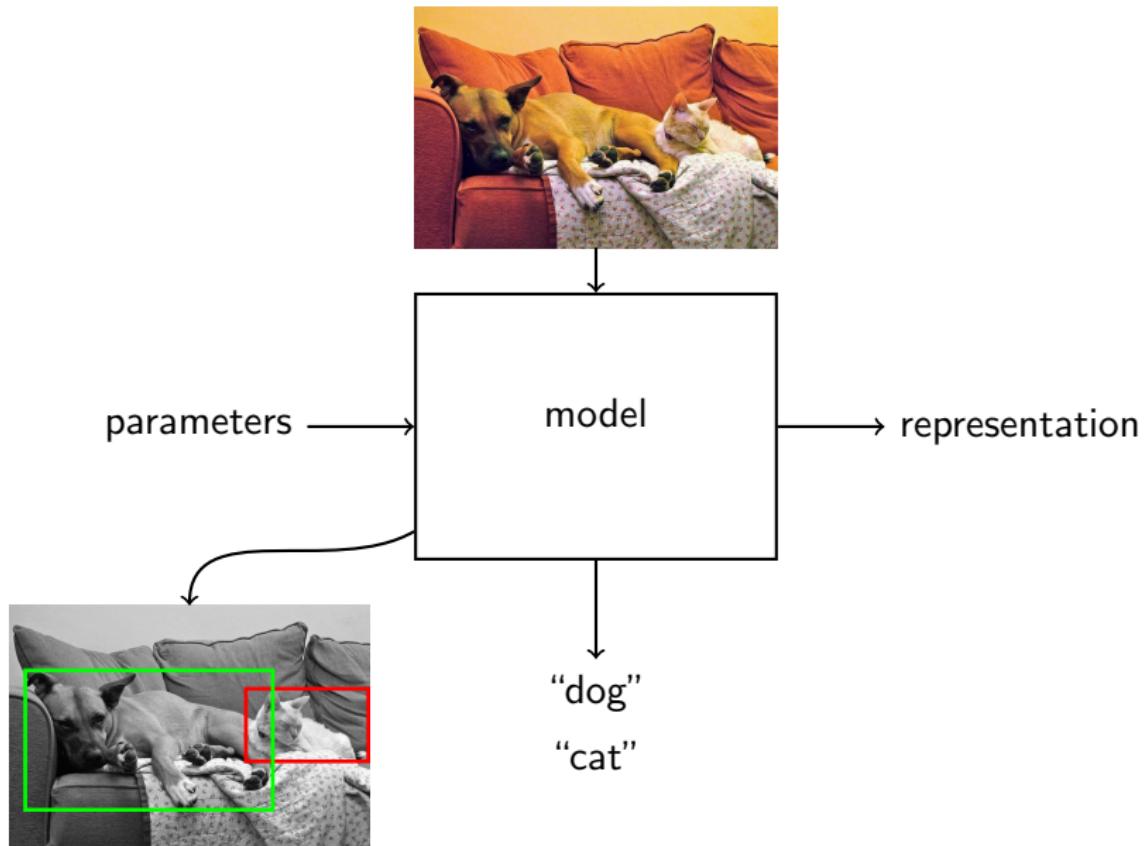
data-driven approach



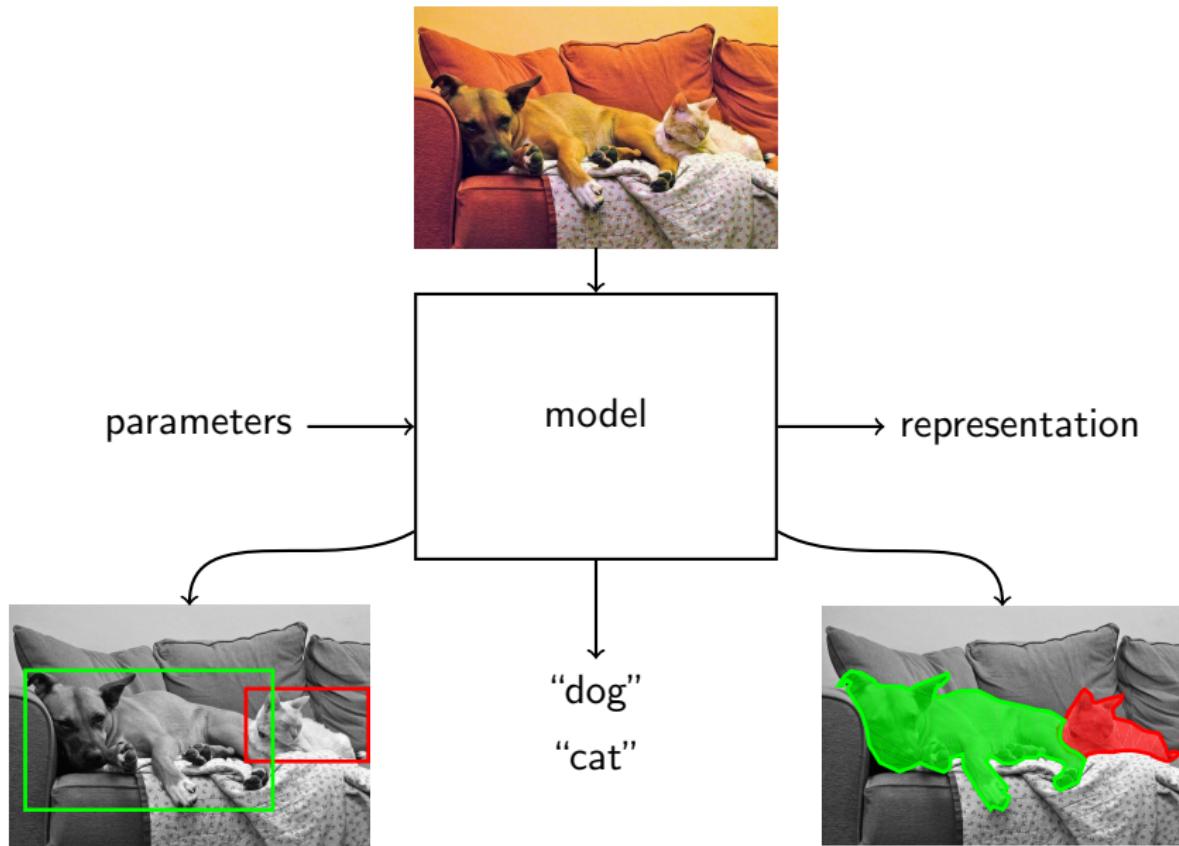
data-driven approach



data-driven approach

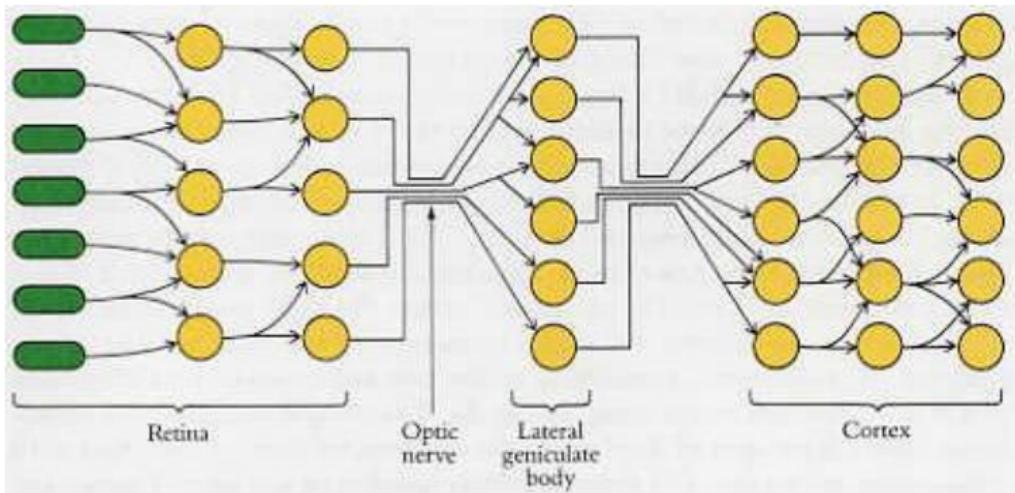


data-driven approach



receptive fields

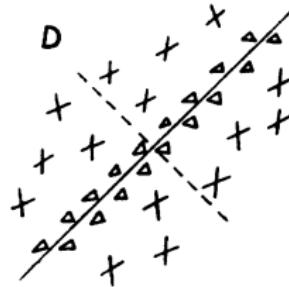
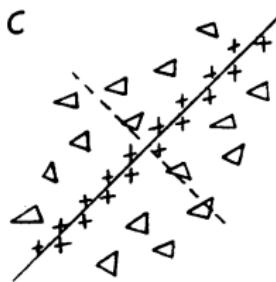
topographic mapping: translation equivariance



- as you move along the retina, the corresponding points in the cortex trace a continuous path
 - each column represents a two-dimensional array of cells
 - a translation in the input causes a translation in the representation

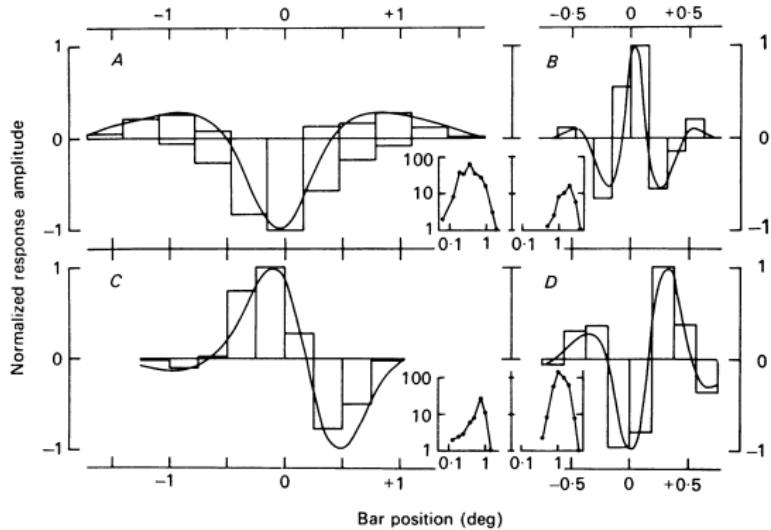
receptive fields

[Hubel and Wiesel 1962]



- A: 'on'-center LGN; B: 'off'-center LGN; C, D: simple cortical
- ×: excitatory ('on'), △: inhibitory ('off') responses
- localized responses, orientation selectivity

linearity



- simple cells perform linear spatial summation over their receptive fields
- spatial response (by oriented bars of varying position)
- frequency response (by oriented gratings of varying frequency)

Movshon, Thompson and Tolhurst. JP 1978. Spatial Summation in the Receptive Fields of Simple Cells in the Cat's Striate Cortex.

linear time-invariant (LTI) systems

- discrete-time signal: $x[n], n \in \mathbb{Z}$
- translation (or shift, or delay): $s_k(x)[n] = x[n - k], k \in \mathbb{Z}$
- linear system (or filter): system commutes with linear combination

$$f\left(\sum_i a_i x_i\right) = \sum_i a_i f(x_i)$$

- time-invariant (or translation equivariant): system commutes with translation

$$f(s_k(x)) = s_k(f(x))$$

linear time-invariant (LTI) systems

- discrete-time signal: $x[n], n \in \mathbb{Z}$
- translation (or shift, or delay): $s_k(x)[n] = x[n - k], k \in \mathbb{Z}$
- linear system (or filter): system commutes with linear combination

$$f \left(\sum_i a_i x_i \right) = \sum_i a_i f(x_i)$$

- time-invariant (or translation equivariant): system commutes with translation

$$f(s_k(x)) = s_k(f(x))$$

convolution

- unit impulse $\delta[n] = \mathbb{1}[n = 0]$
- every signal x expressed as

$$x[n] = \sum_k x[k]\delta[n - k] = \sum_k x[k]s_k(\delta)[n]$$

- if f is LTI with impulse response $h = f(\delta)$, then $f(x) = x * h$:

$$\begin{aligned}f(x)[n] &= f\left(\sum_k x[k]s_k(\delta)\right)[n] = \sum_k x[k]s_k(f(\delta))[n] \\&= \sum_k x[k]h[n - k] = x * h[n]\end{aligned}$$

- Q: what is $\delta * h$ for any h ? what is $s_k(\delta) * h$?

convolution

- unit impulse $\delta[n] = \mathbb{1}[n = 0]$
- every signal x expressed as

$$x[n] = \sum_k x[k]\delta[n - k] = \sum_k x[k]s_k(\delta)[n]$$

- if f is LTI with impulse response $h = f(\delta)$, then $f(x) = x * h$:

$$\begin{aligned}f(x)[n] &= f\left(\sum_k x[k]s_k(\delta)\right)[n] = \sum_k x[k]s_k(f(\delta))[n] \\&= \sum_k x[k]h[n - k] := (x * h)[n]\end{aligned}$$

- Q: what is $\delta * h$ for any h ? what is $s_k(\delta) * h$?

convolution

- unit impulse $\delta[n] = \mathbb{1}[n = 0]$
- every signal x expressed as

$$x[n] = \sum_k x[k]\delta[n - k] = \boxed{\sum_k x[k]s_k(\delta)[n]}$$

- if f is LTI with impulse response $h = f(\delta)$, then $f(x) = x * h$:

$$\begin{aligned} f(x)[n] &= f \left(\sum_k x[k]s_k(\delta) \right) [n] = \sum_k x[k]s_k(f(\delta))[n] \\ &= \sum_k x[k]h[n - k] := (x * h)[n] \end{aligned}$$

- Q: what is $\delta * h$ for any h ? what is $s_k(\delta) * h$?

convolution

- unit impulse $\delta[n] = \mathbb{1}[n = 0]$
- every signal x expressed as

$$x[n] = \sum_k x[k]\delta[n - k] = \sum_k x[k]s_k(\delta)[n]$$

- if f is LTI with impulse response $h = f(\delta)$, then $f(x) = x * h$:

$$\begin{aligned} f(x)[n] &= \boxed{f} \left(\sum_k x[k]s_k(\delta) \right) [n] = \sum_k x[k]s_k(\boxed{f(\delta)})[n] \\ &= \sum_k x[k]h[n - k] := (x * h)[n] \end{aligned}$$

- Q: what is $\delta * h$ for any h ? what is $s_k(\delta) * h$?

convolution

- unit impulse $\delta[n] = \mathbb{1}[n = 0]$
- every signal x expressed as

$$x[n] = \sum_k x[k]\delta[n - k] = \sum_k x[k]s_k(\delta)[n]$$

- if f is LTI with impulse response $h = f(\delta)$, then $f(x) = x * h$:

$$\begin{aligned} f(x)[n] &= f\left(\sum_k x[k]s_k(\delta)\right)[n] = \sum_k x[k]s_k(f(\delta))[n] \\ &= \sum_k x[k]h[n - k] := (x * h)[n] \end{aligned}$$

- Q: what is $\delta * h$ for any h ? what is $s_k(\delta) * h$?

convolution

- unit impulse $\delta[n] = \mathbb{1}[n = 0]$
- every signal x expressed as

$$x[n] = \sum_k x[k]\delta[n - k] = \sum_k x[k]s_k(\delta)[n]$$

- if f is LTI with impulse response $h = f(\delta)$, then $f(x) = x * h$:

$$\begin{aligned} f(x)[n] &= f\left(\sum_k x[k]s_k(\delta)\right)[n] = \sum_k x[k]s_k(f(\delta))[n] \\ &= \sum_k x[k]h[n - k] = (x * h)[n] \end{aligned}$$

- Q: what is $\delta * h$ for any h ? what is $s_k(\delta) * h$?

convolution

- unit impulse $\delta[n] = \mathbb{1}[n = 0]$
- every signal x expressed as

$$x[n] = \sum_k x[k]\delta[n - k] = \sum_k x[k]s_k(\delta)[n]$$

- if f is LTI with impulse response $h = f(\delta)$, then $f(x) = x * h$:

$$\begin{aligned}f(x)[n] &= f\left(\sum_k x[k]s_k(\delta)\right)[n] = \sum_k x[k]s_k(f(\delta))[n] \\&= \sum_k x[k]h[n - k] := (x * h)[n]\end{aligned}$$

- Q: what is $\delta * h$ for any h ? what is $s_k(\delta) * h$?

convolution

- unit impulse $\delta[n] = \mathbb{1}[n = 0]$
- every signal x expressed as

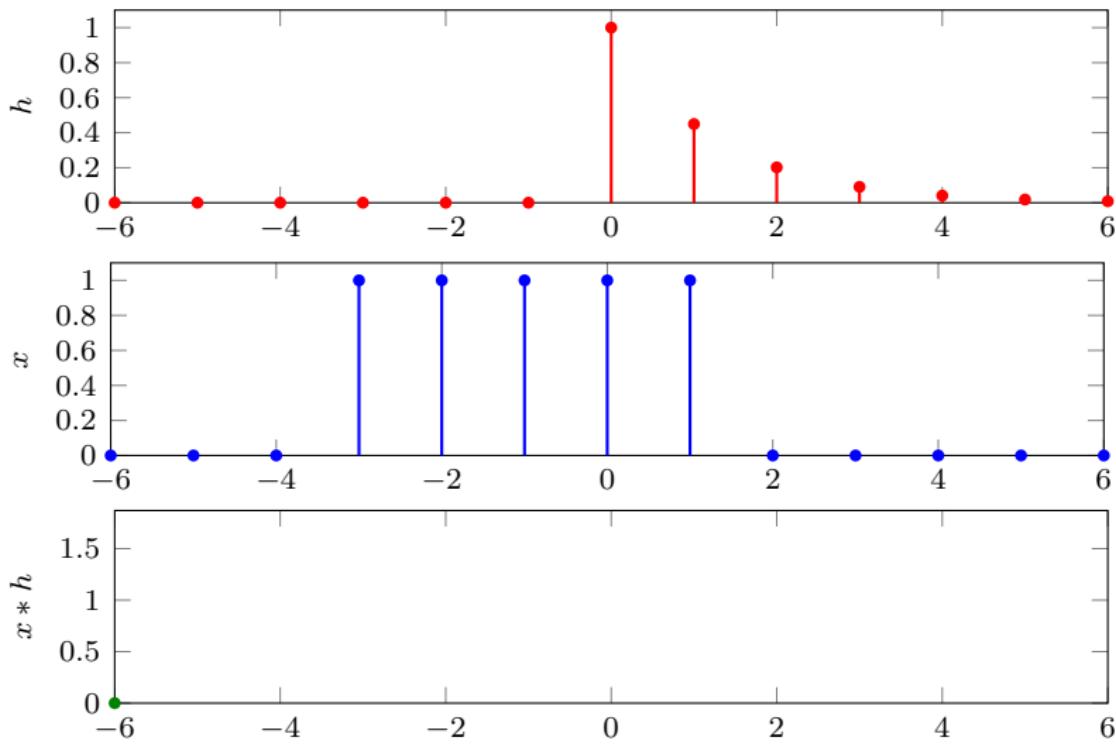
$$x[n] = \sum_k x[k]\delta[n - k] = \sum_k x[k]s_k(\delta)[n]$$

- if f is LTI with impulse response $h = f(\delta)$, then $f(x) = x * h$:

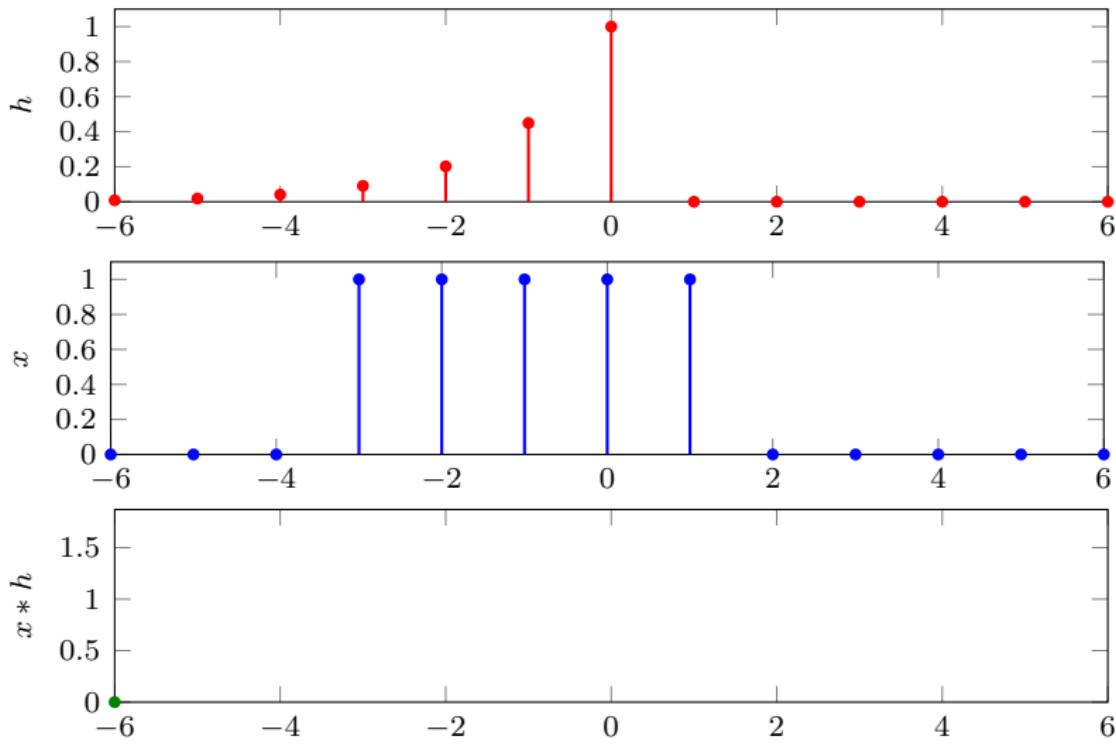
$$\begin{aligned} f(x)[n] &= f\left(\sum_k x[k]s_k(\delta)\right)[n] = \sum_k x[k]s_k(f(\delta))[n] \\ &= \sum_k x[k]h[n - k] := (x * h)[n] \end{aligned}$$

- Q: what is $\delta * h$ for any h ? what is $s_k(\delta) * h$?

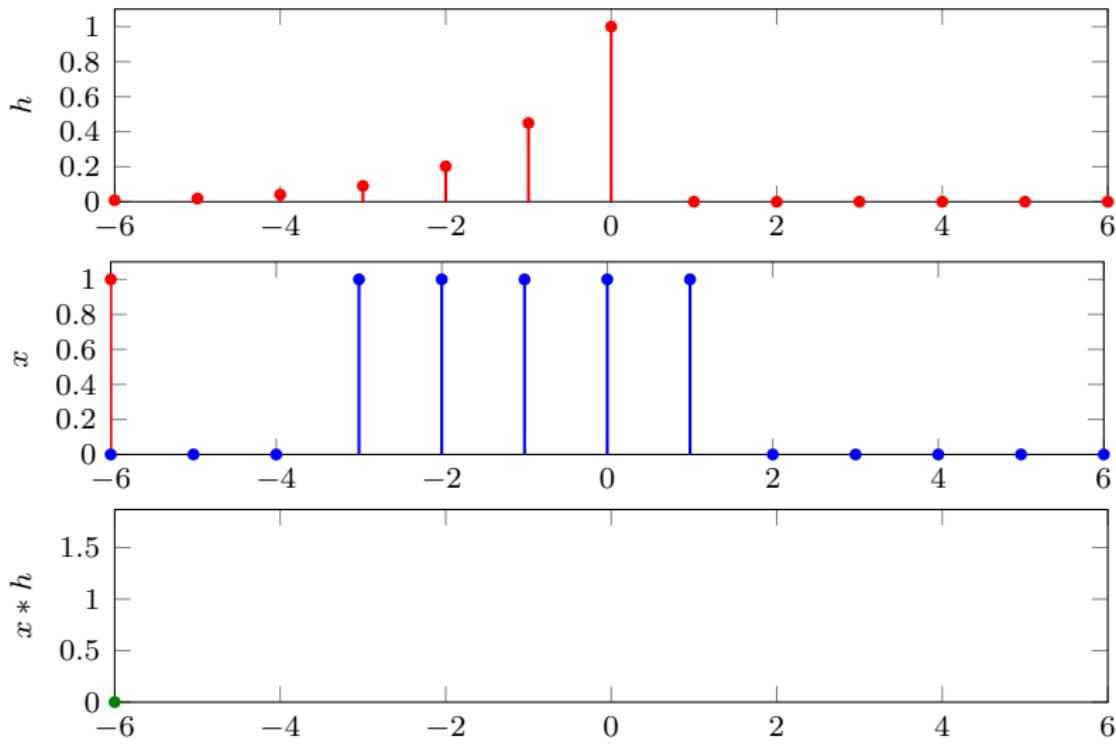
convolution



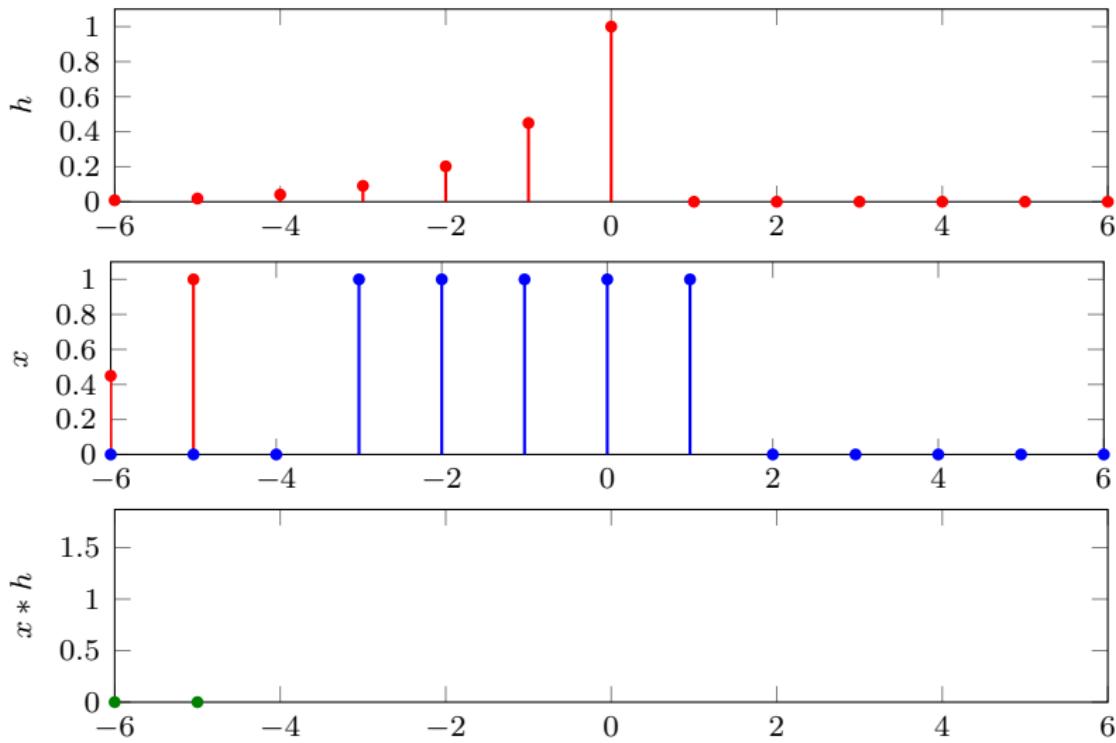
convolution



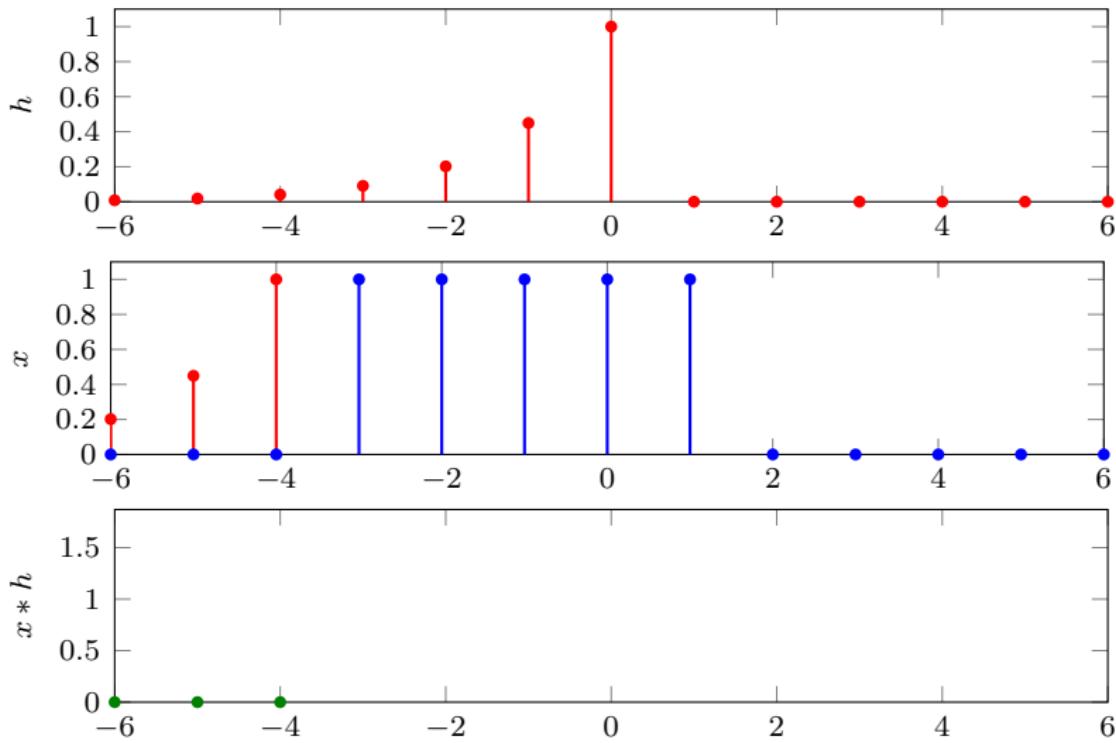
convolution



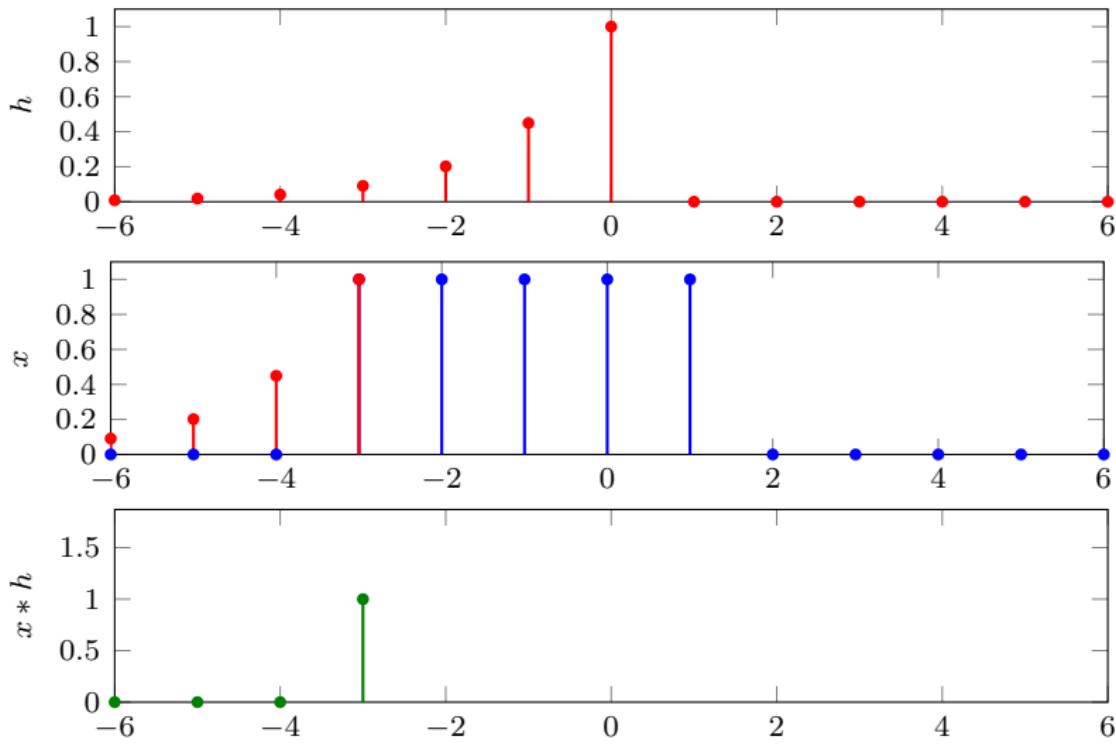
convolution



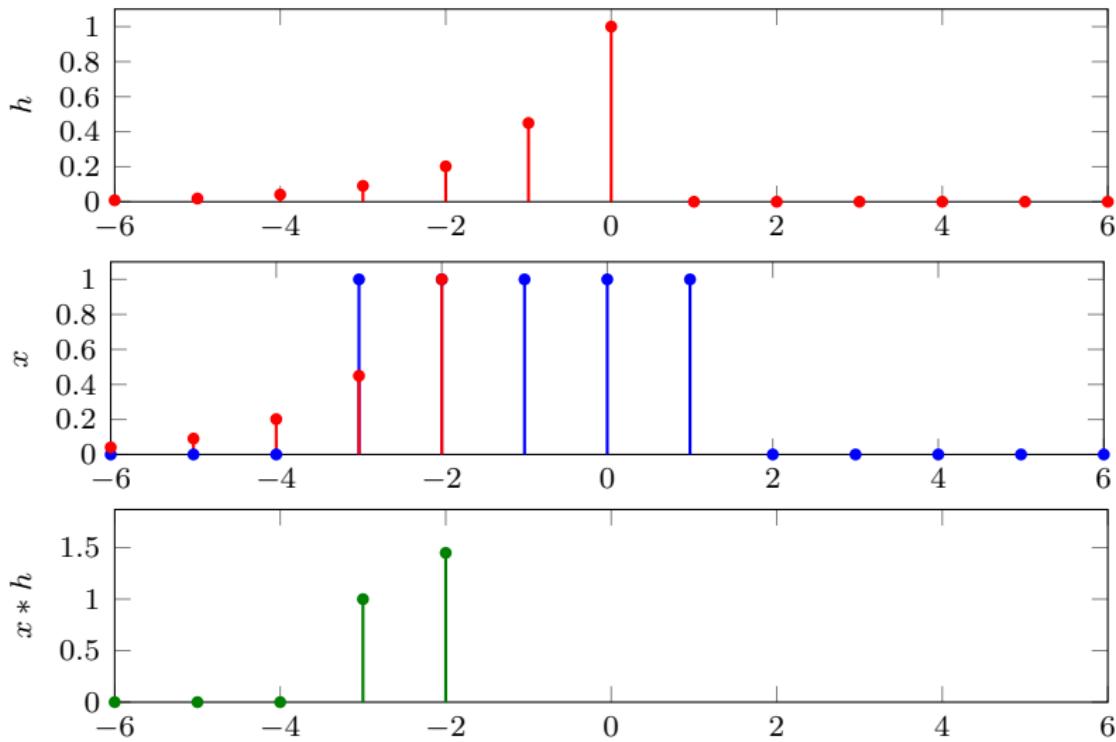
convolution



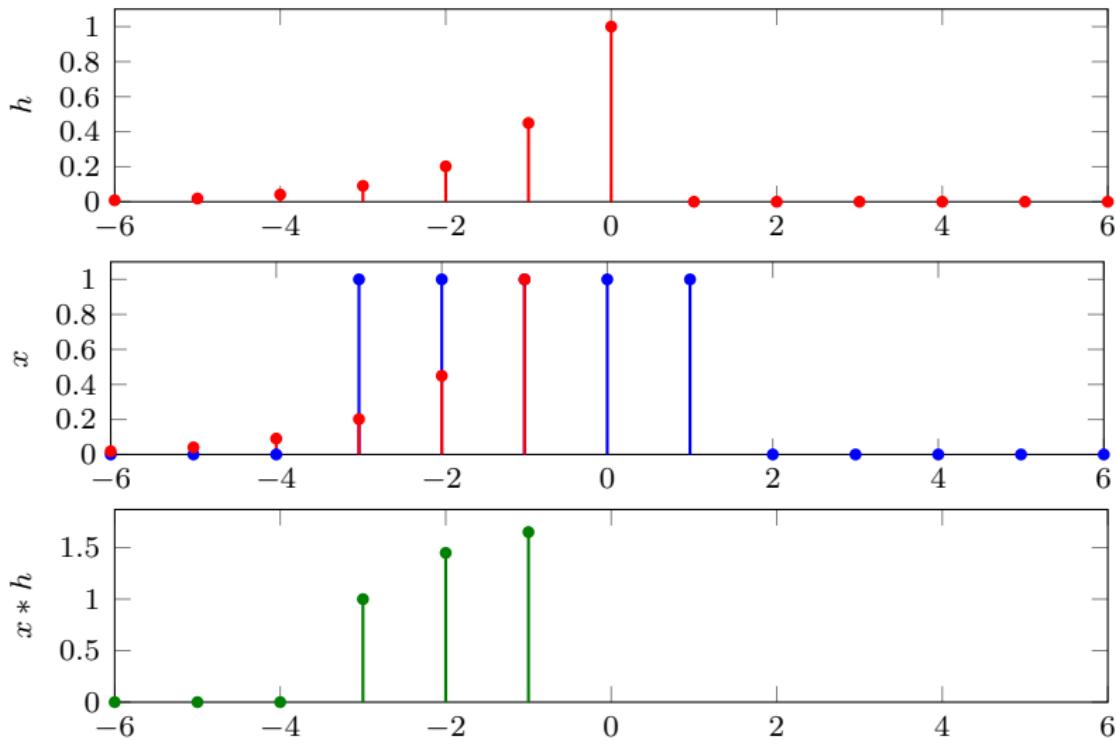
convolution



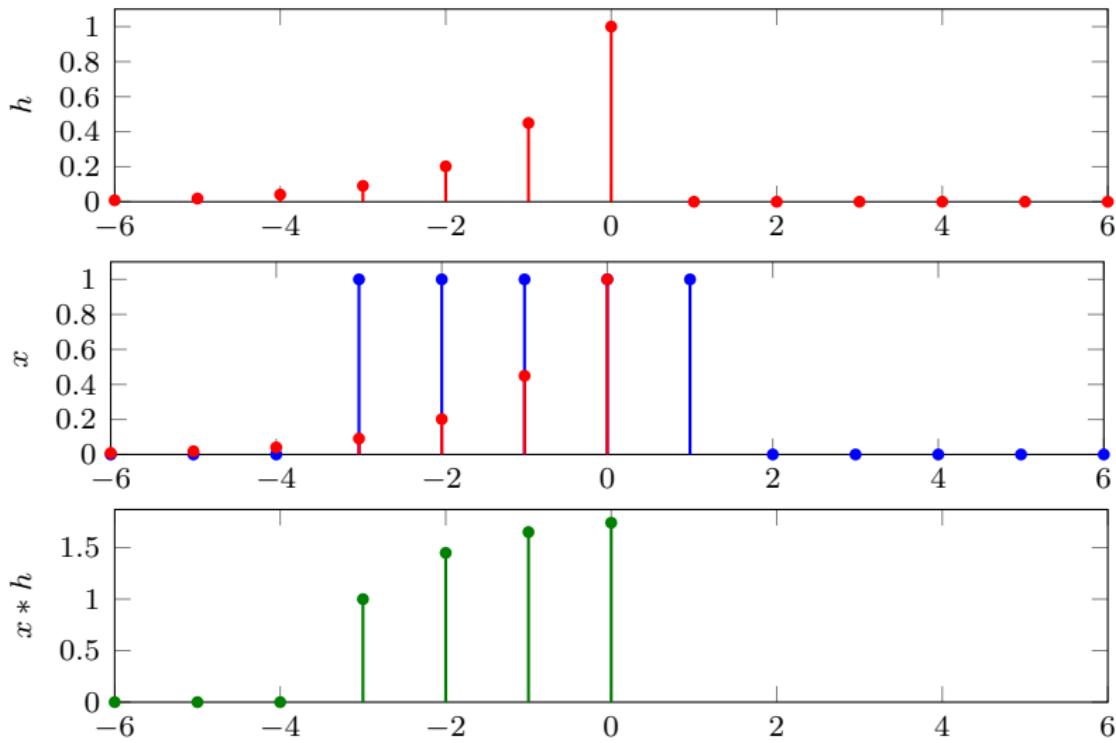
convolution



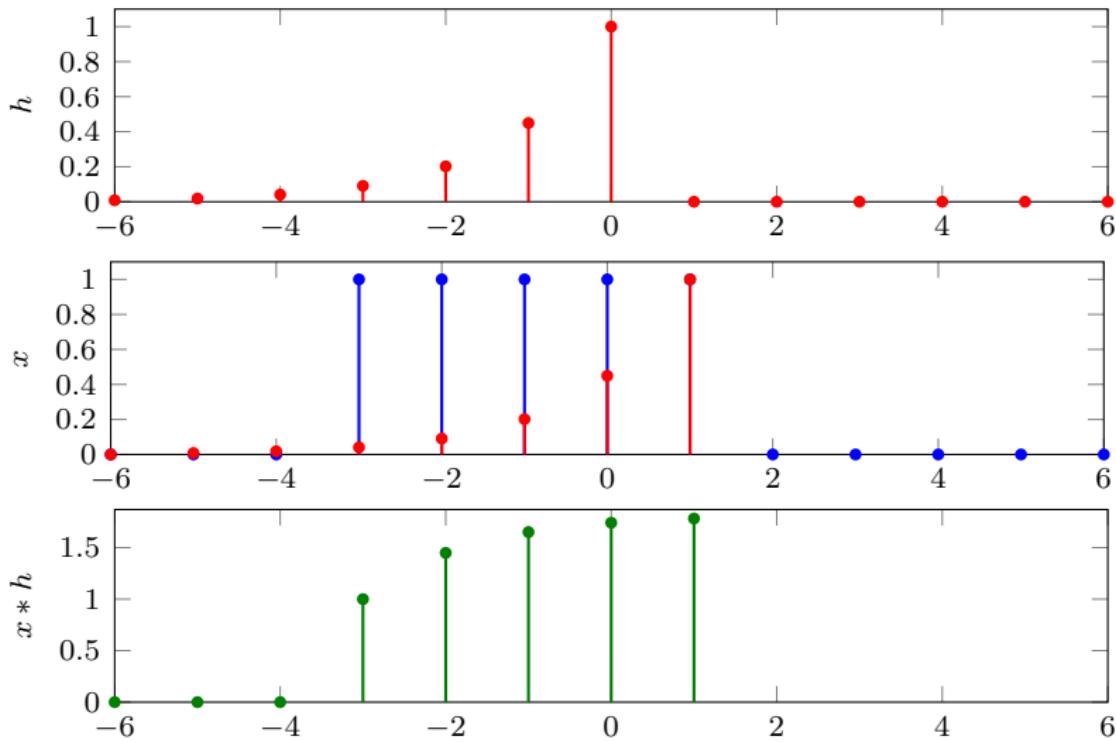
convolution



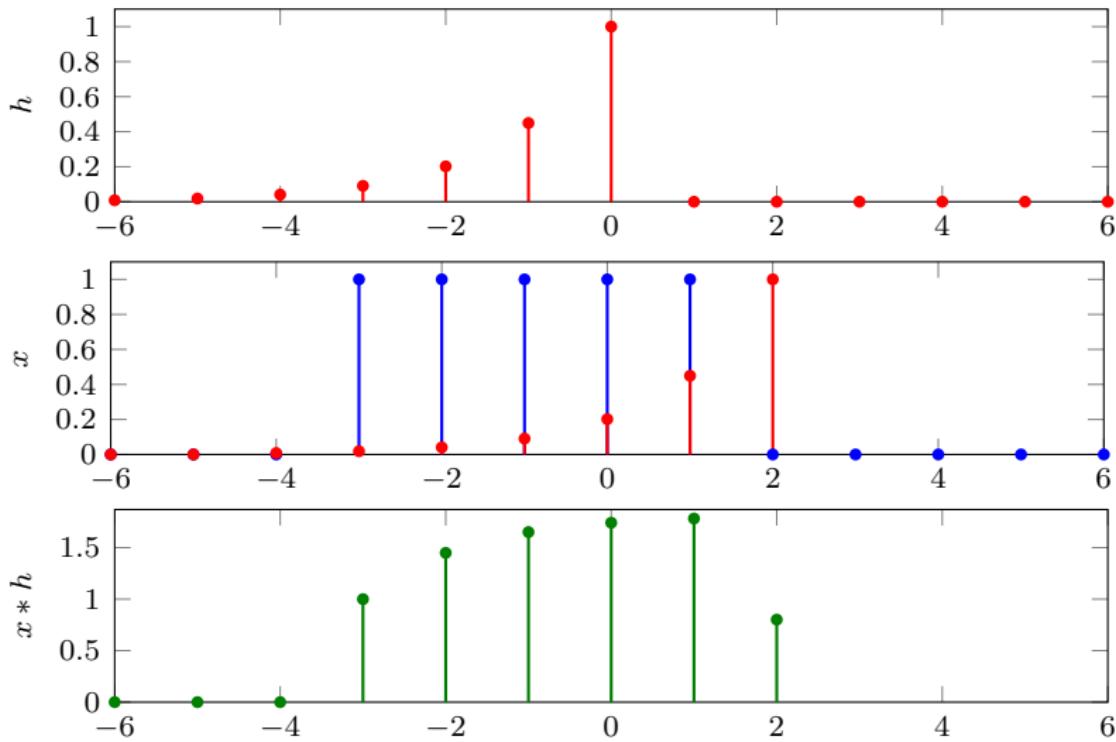
convolution



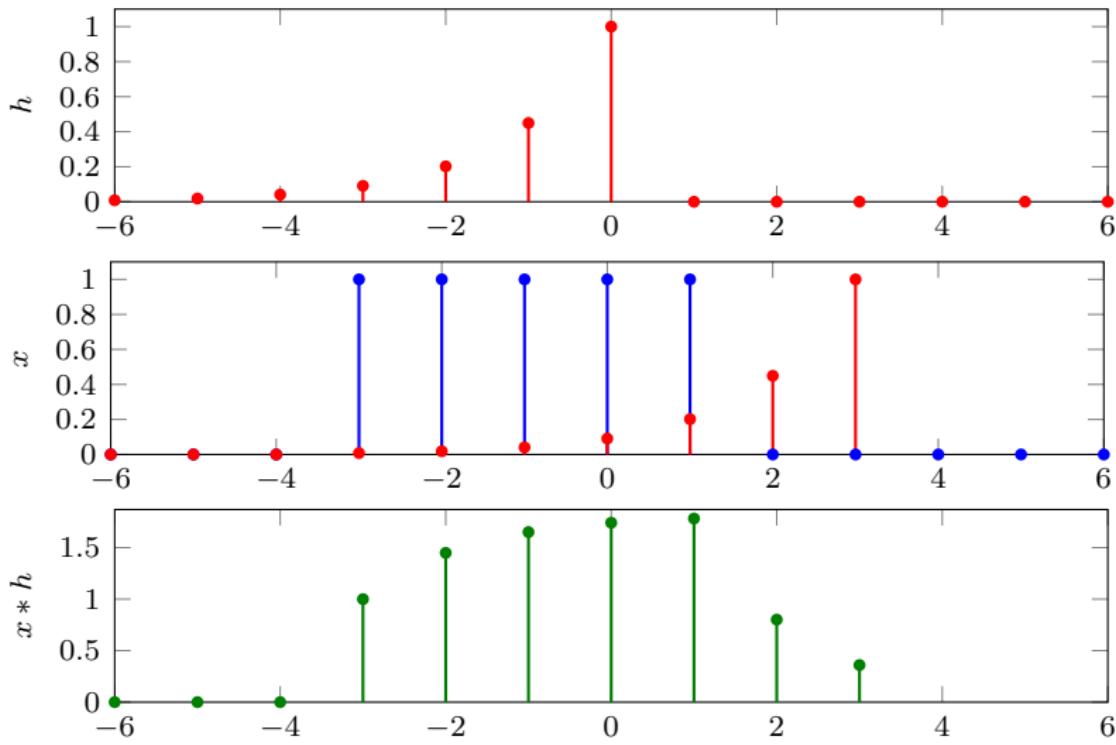
convolution



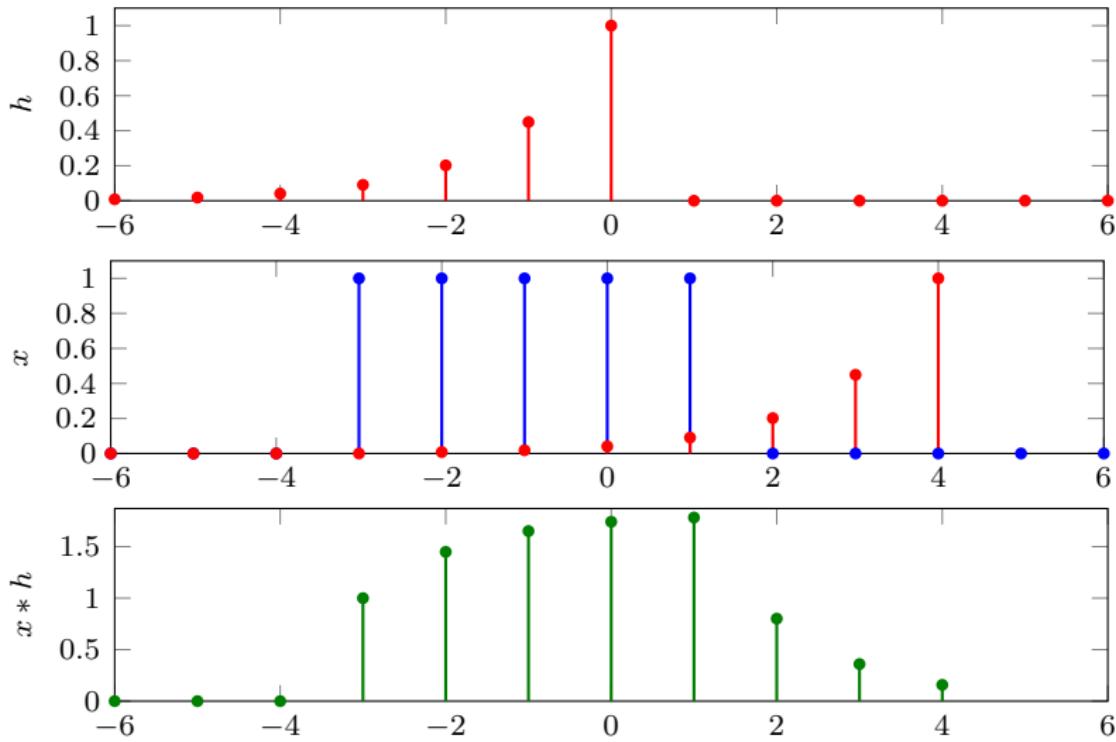
convolution



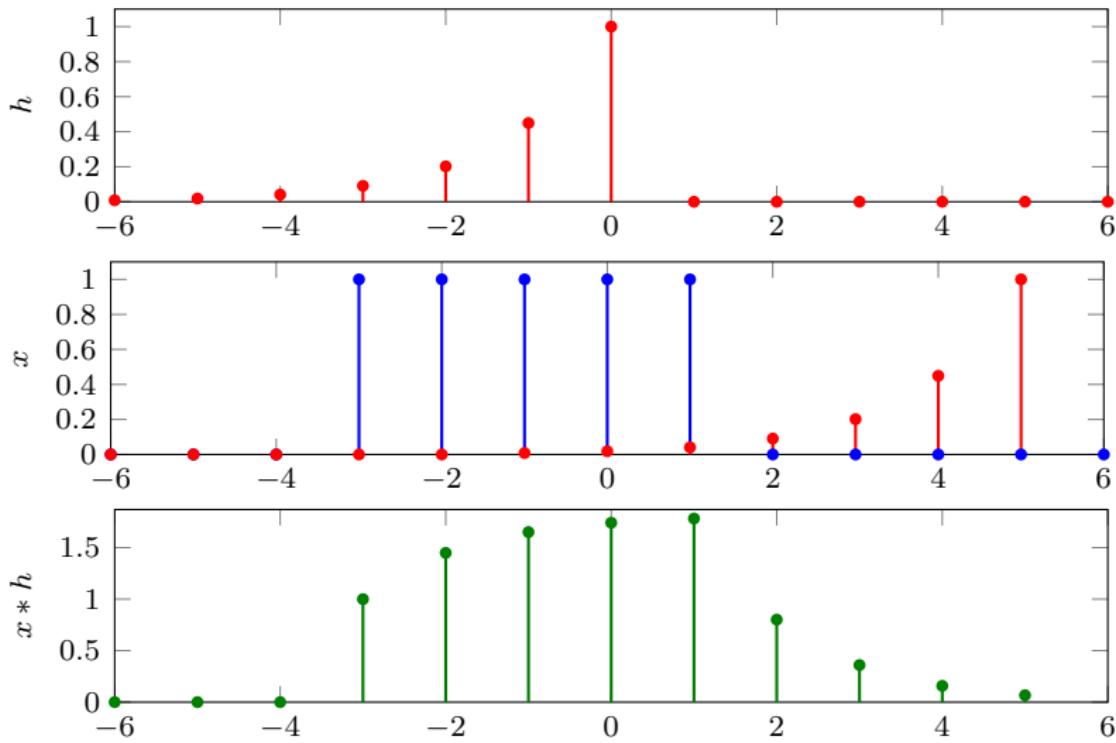
convolution



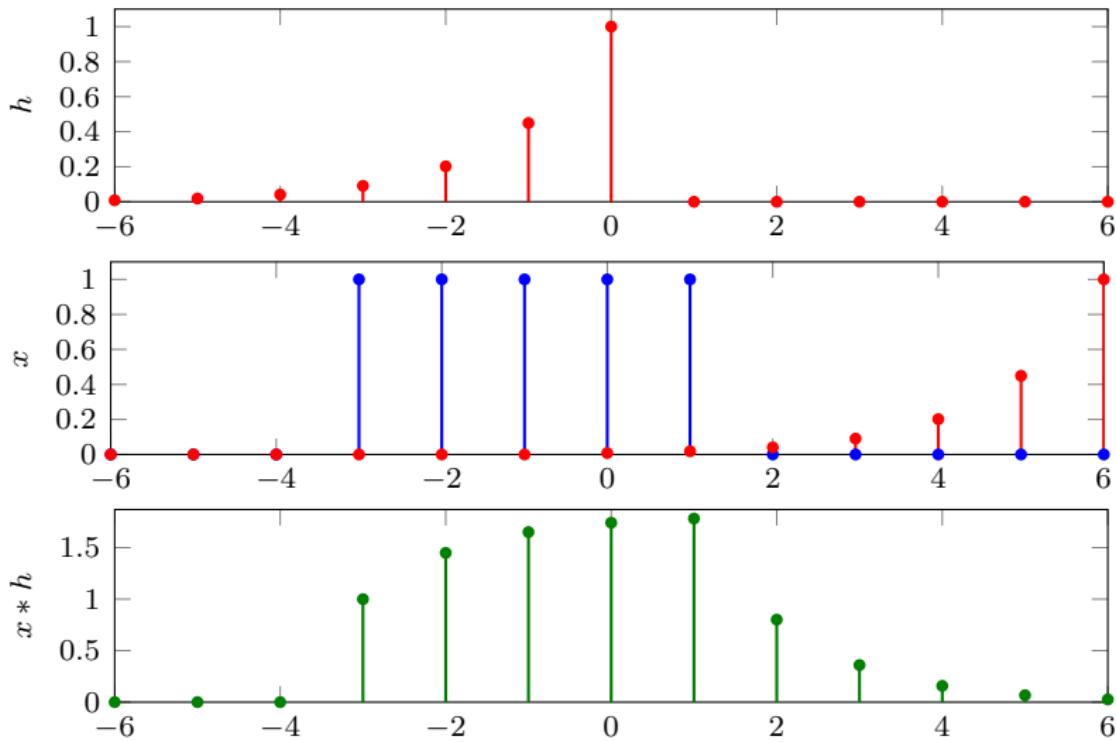
convolution



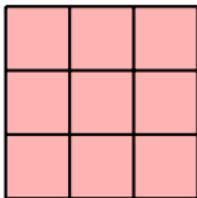
convolution



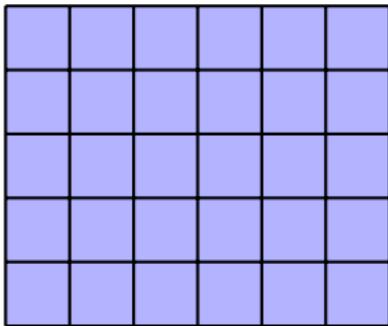
convolution



2d convolution

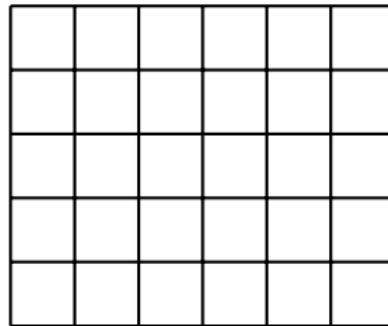


h



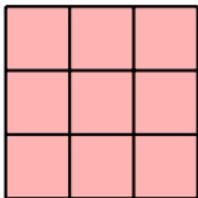
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

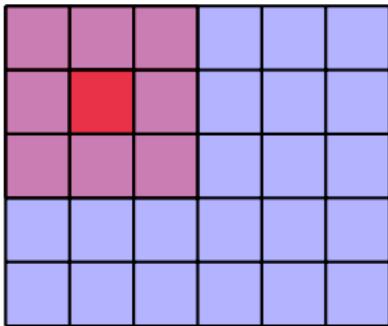


$x * h$

2d convolution

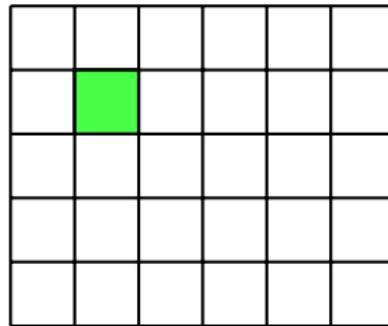


h



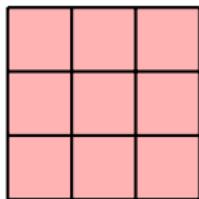
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

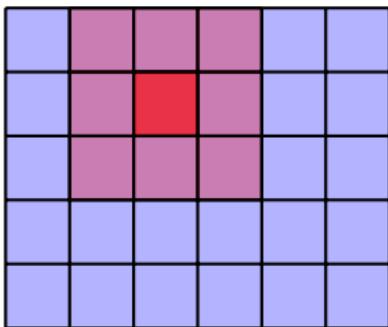


$x * h$

2d convolution

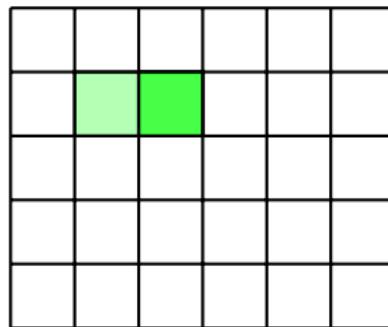


h



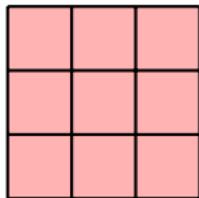
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

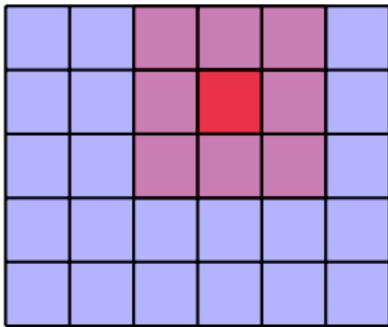


$x * h$

2d convolution

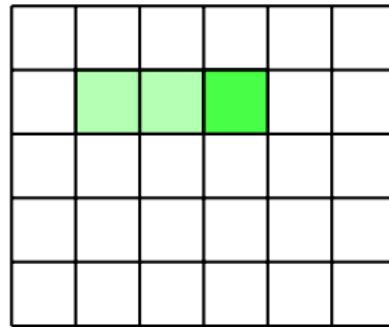


h



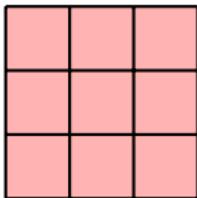
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

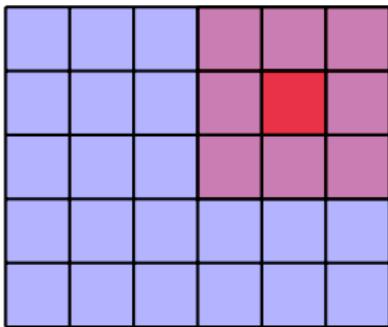


$x * h$

2d convolution

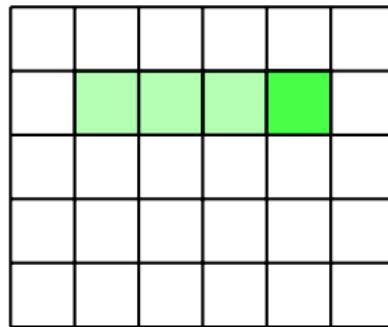


h



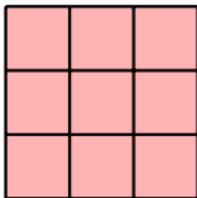
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

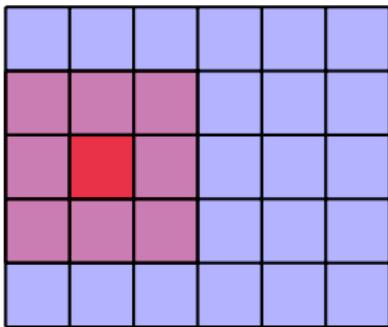


$x * h$

2d convolution

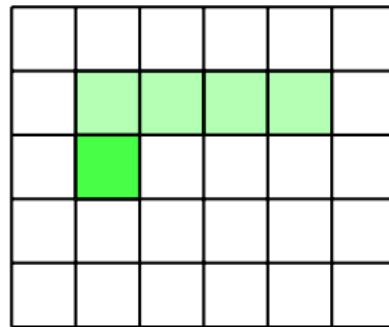


h



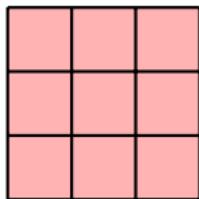
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

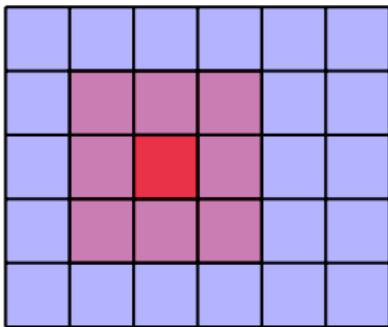


$x * h$

2d convolution

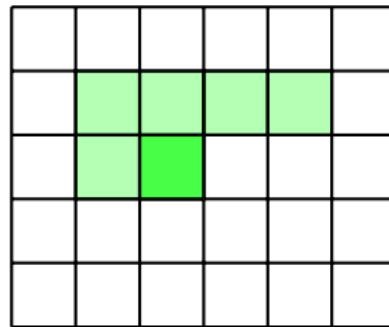


h



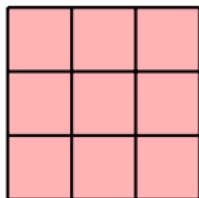
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

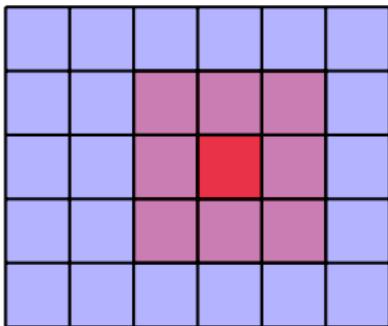


$x * h$

2d convolution

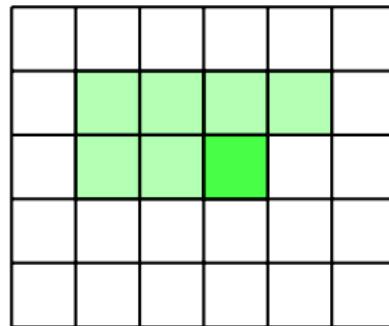


h



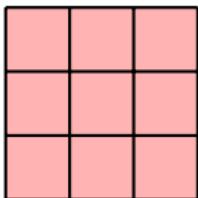
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

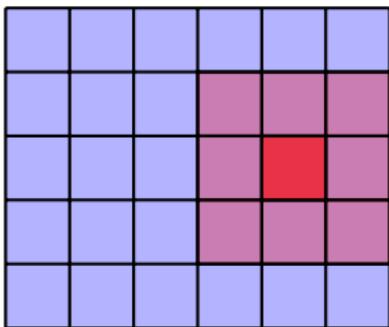


$x * h$

2d convolution

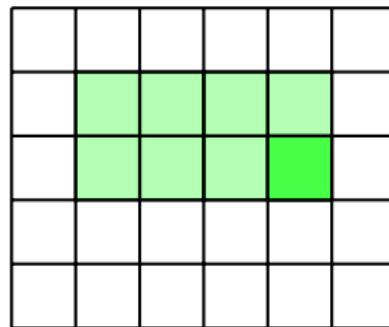


h



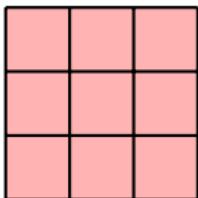
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

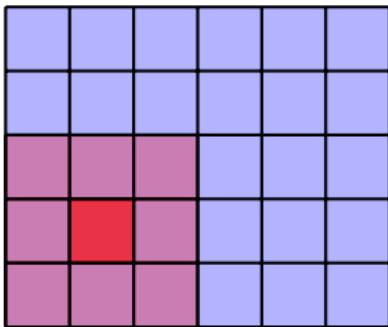


$x * h$

2d convolution

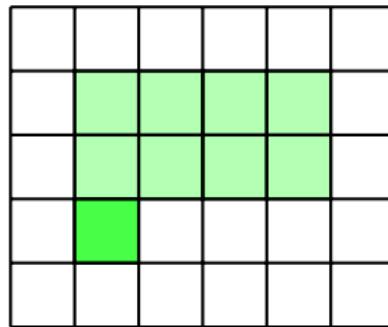


h



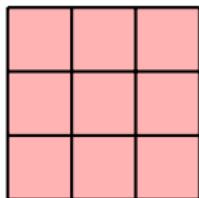
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

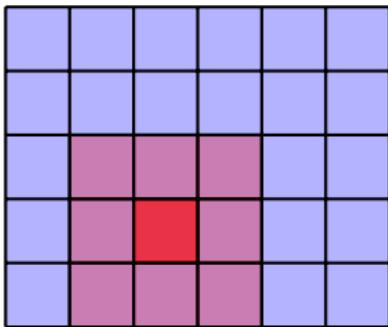


$x * h$

2d convolution

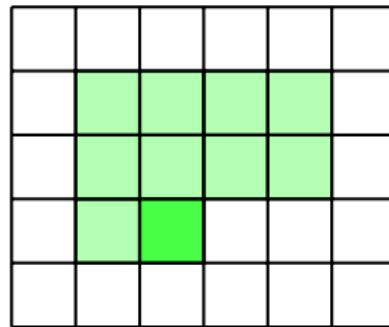


h



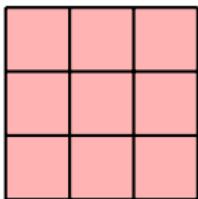
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

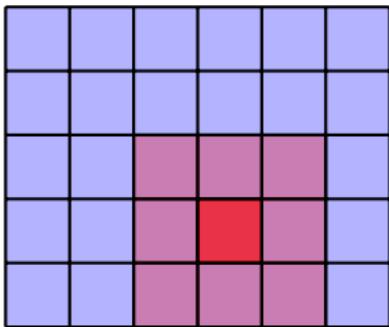


$x * h$

2d convolution

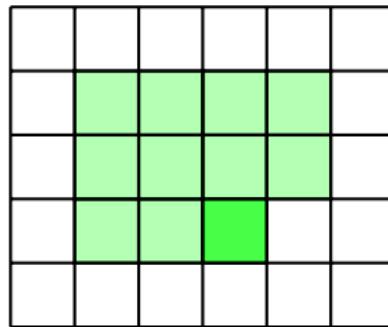


h



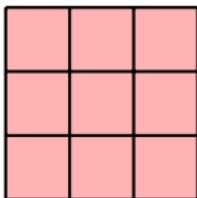
x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$

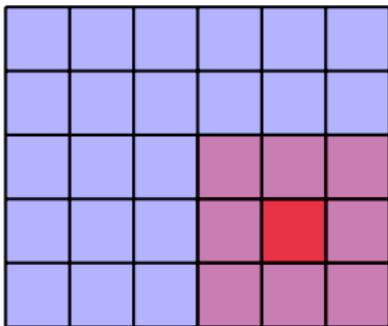


$x * h$

2d convolution

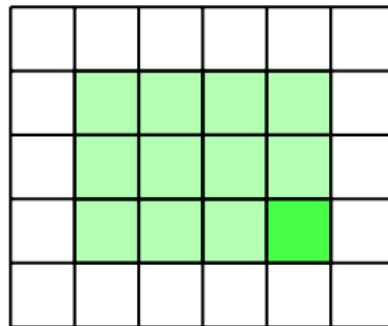


h



x

$$(x * h)[\mathbf{n}] = \sum_{\mathbf{k}} h[\mathbf{k}]x[\mathbf{n} - \mathbf{k}]$$



$x * h$

continuous time

- continuous-time signal: $x(t)$, $t \in \mathbb{R}$
- translation (or shift, or delay): $s_\tau(x)(t) = x(t - \tau)$, $\tau \in \mathbb{R}$
- LTI system definition: same
- Dirac delta “function” δ : every signal x expressed as

$$x(t) = \int x(\tau) \delta(t - \tau) d\tau$$

- convolution: f LTI, impulse response $h = f(\delta)$ implies

$$f(x)(t) = (x * h)(t) := \int x(\tau) h(t - \tau) d\tau$$

continuous time

- continuous-time signal: $x(t)$, $t \in \mathbb{R}$
- translation (or shift, or delay): $s_\tau(x)(t) = x(t - \tau)$, $\tau \in \mathbb{R}$
- LTI system definition: same
- Dirac delta “function” δ : every signal x expressed as

$$x(t) = \int x(\tau) \delta(t - \tau) d\tau$$

- convolution: f LTI, impulse response $h = f(\delta)$ implies

$$f(x)(t) = (x * h)(t) := \int x(\tau) h(t - \tau) d\tau$$

continuous time

- continuous-time signal: $x(t)$, $t \in \mathbb{R}$
- translation (or shift, or delay): $s_\tau(x)(t) = x(t - \tau)$, $\tau \in \mathbb{R}$
- LTI system definition: same
- Dirac delta “function” δ : every signal x expressed as

$$x(t) = \int x(\tau) \delta(t - \tau) d\tau$$

- convolution: f LTI, impulse response $h = f(\delta)$ implies

$$f(x)(t) = (x * h)(t) := \int x(\tau) h(t - \tau) d\tau$$

Fourier transform

- time (or space) → frequency

$$X(f) = \int x(t)e^{-j2\pi ft} dt$$

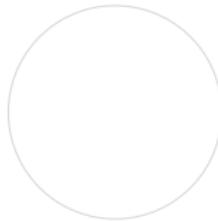
- frequency → time (or space)

$$x(t) = \int X(f)e^{-j2\pi ft} df$$

- measurements



bar (+)



bar (-)



grating

Fourier transform

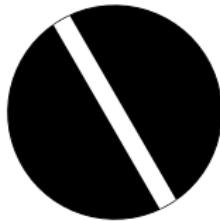
- time (or space) → frequency

$$X(f) = \int x(t)e^{-j2\pi ft} dt$$

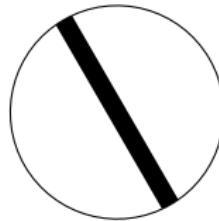
- frequency → time (or space)

$$x(t) = \int X(f)e^{-j2\pi ft} df$$

- measurements



bar (+)



bar (-)



grating

Fourier transform

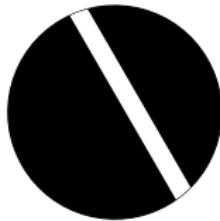
- time (or space) → frequency

$$X(f) = \int x(t)e^{-j2\pi ft} dt$$

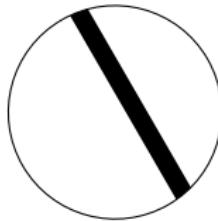
- frequency → time (or space)

$$x(t) = \int X(f)e^{-j2\pi ft} df$$

- measurements



bar (+)



bar (-)



grating

Fourier transform

- time (or space) → frequency

$$X(f) = \int x(t)e^{-j2\pi ft} dt$$

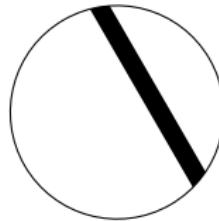
- frequency → time (or space)

$$x(t) = \int X(f)e^{-j2\pi ft} df$$

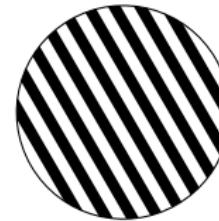
- measurements



bar (+)

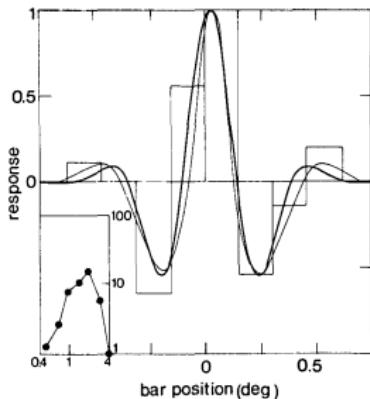


bar (-)



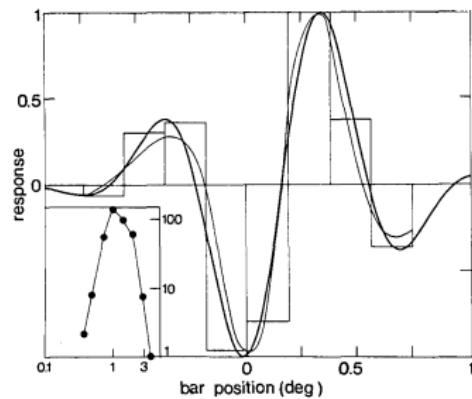
grating

mathematical model



symmetric

$$e^{-a^2(x-x_0)^2} \cos(2\pi f_0(x - x_0))$$

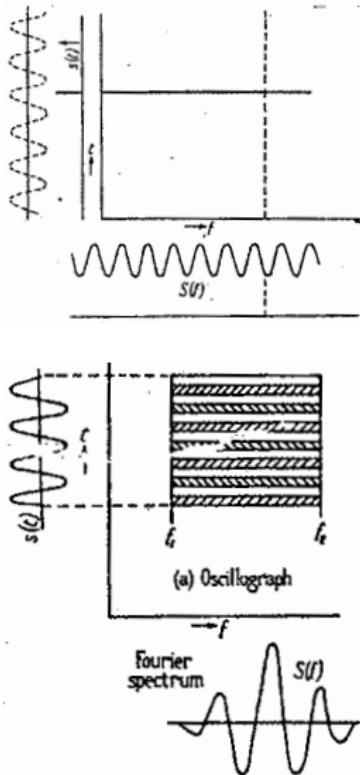


antisymmetric

$$e^{-a^2(x-x_0)^2} \sin(2\pi f_0(x - x_0))$$

- (thin) experimental: inverse Fourier of grating stimuli responses
- (thick) least-squares fit of Gabor elementary signal

Gabor elementary signals



- “effective duration”

$$\Delta t = [2\pi(t - \bar{t})^2]^{1/2}$$

- “effective bandwidth”

$$\Delta f = [2\pi(f - \bar{f})^2]^{1/2}$$

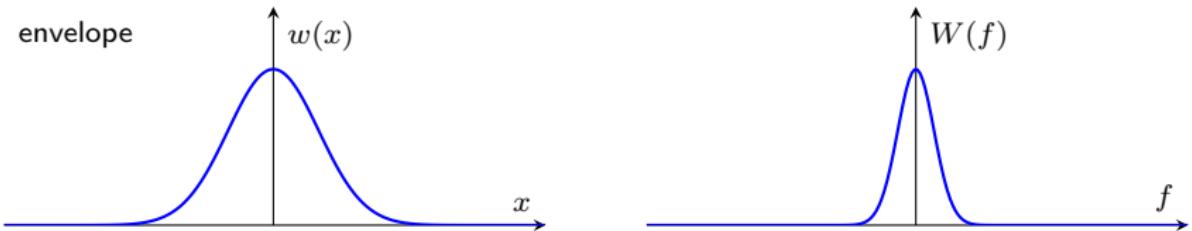
- uncertainty principle

$$\Delta t \Delta f \geq \frac{1}{2}$$

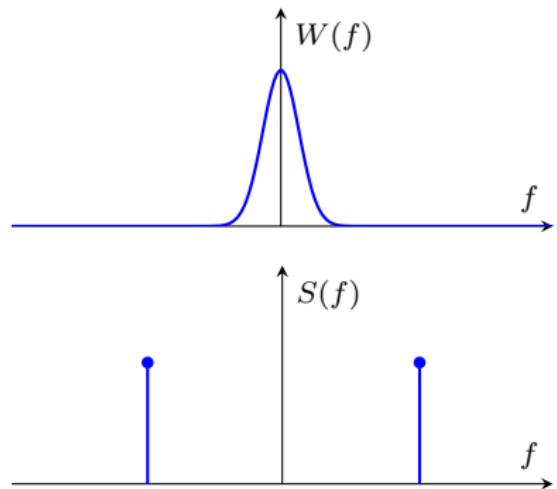
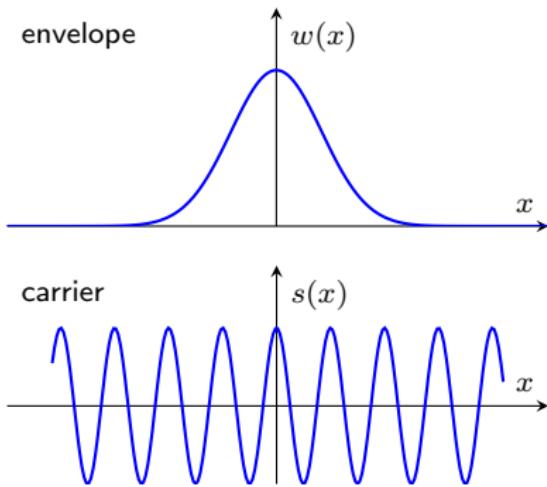
- minimal solution

$$\psi(t) = e^{-a^2(t-t_0)^2} e^{j2\pi f_0(t-t_0)}$$

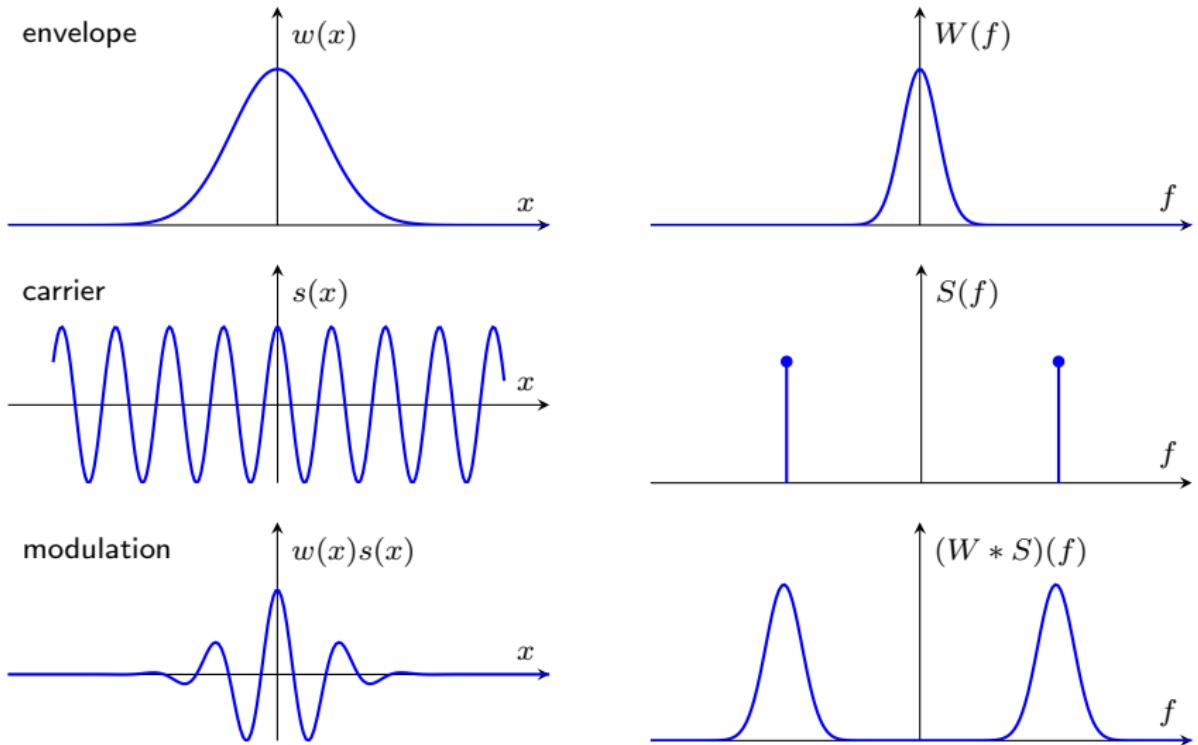
convolution theorem & modulation



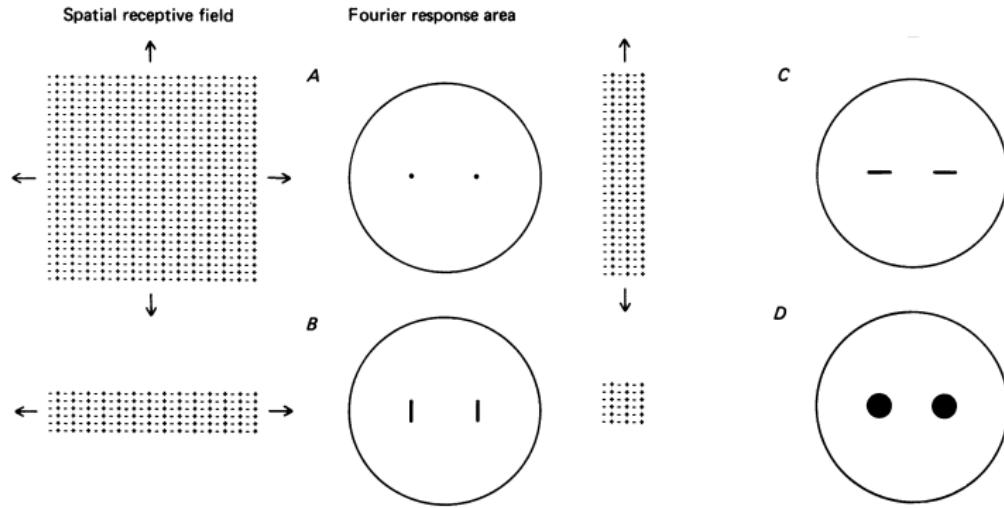
convolution theorem & modulation



convolution theorem & modulation



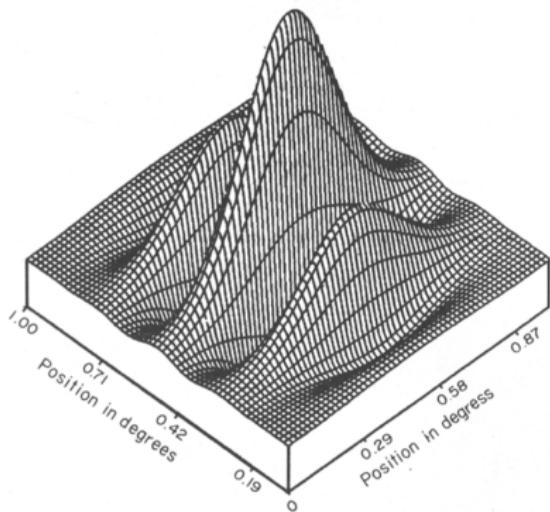
2d space/frequency considerations



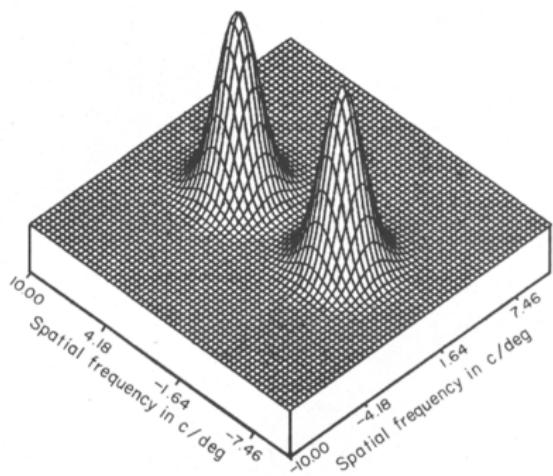
- responses to gratings at different frequencies and orientations
 - localized in space and frequency, in both dimensions

2d space/frequency considerations

(a) Excitability profile

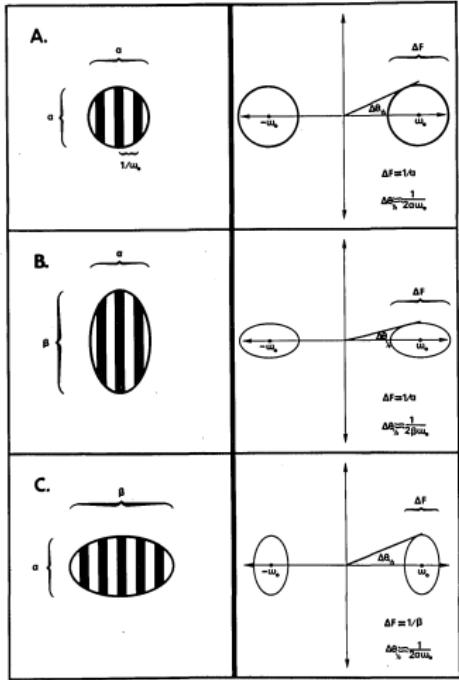


(b) 2-D Fourier transform of profile



- spatial frequency and orientation are separable
- by inverse Fourier, hypothesize a 2d spatial 'receptive field profile'

2d Gabor filters



- 2d uncertainty principle

$$\Delta x \Delta u \geq \frac{1}{4}$$

- minimal solution

$$f(\mathbf{x}) = e^{-\pi w_{\mathbf{x}_0, A}(\mathbf{x})} e^{j2\pi c_{\mathbf{x}_0, \mathbf{u}_0}(\mathbf{x})}$$

$$F(\mathbf{u}) = e^{-\pi w_{\mathbf{u}_0, A^{-1}}(\mathbf{u})} e^{j2\pi c_{\mathbf{u}_0, \mathbf{x}_0}(\mathbf{u})}$$

- envelope & carrier signals

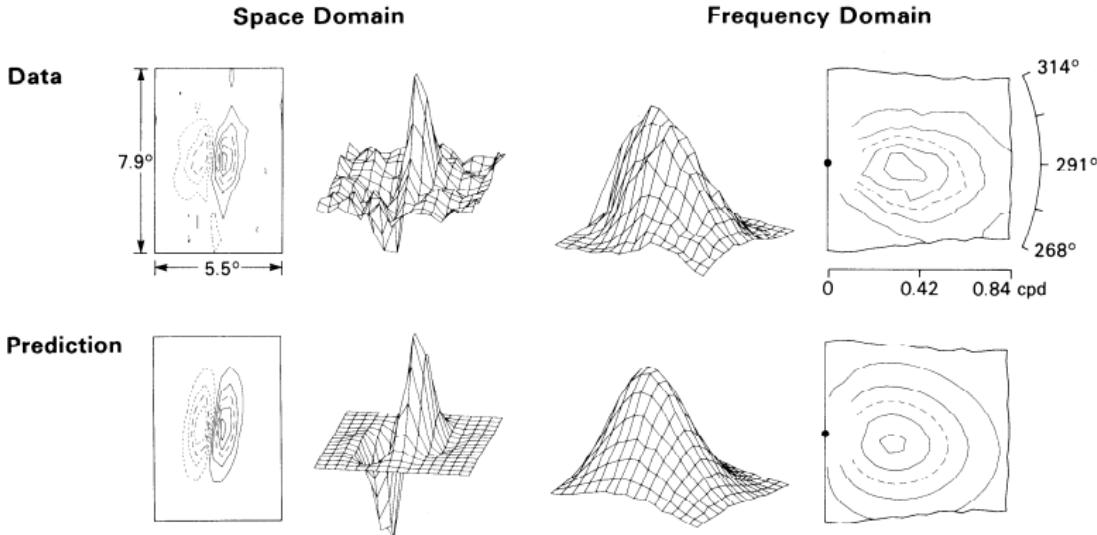
$$w_{\mathbf{x}_0, A}(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_0)^\top A^2 (\mathbf{x} - \mathbf{x}_0)$$

$$c_{\mathbf{x}_0, \mathbf{u}_0}(\mathbf{x}) = \mathbf{u}_0^\top (\mathbf{x} - \mathbf{x}_0)$$

$$A = \text{diag}(a, b)$$

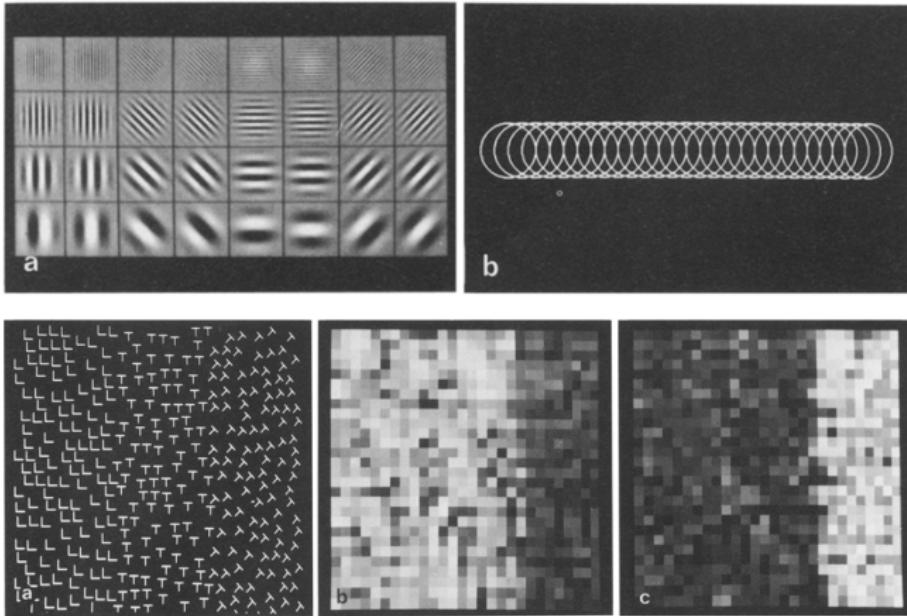
Daugman. JOSA 1985. Uncertainty Relation for Resolution in Space, Spatial Frequency, and Orientation Optimized By Two-Dimensional Visual Cortical Filters.

Gabor hypothesis verified



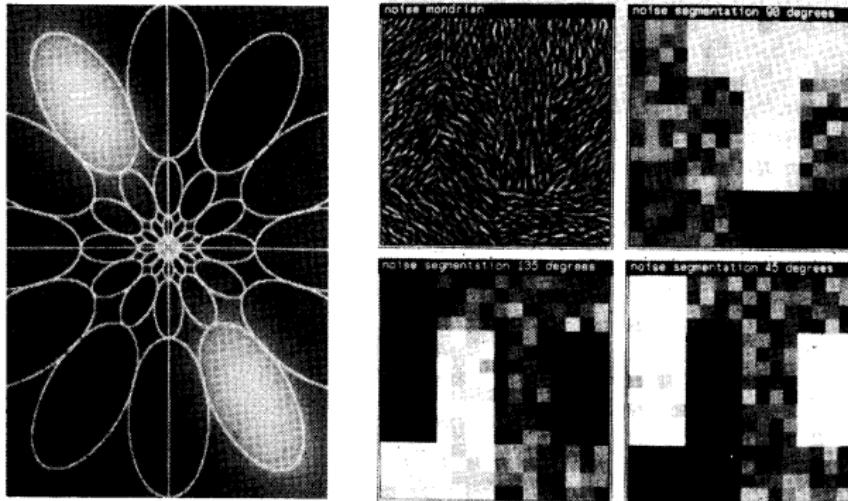
- compare spatial data to Gabor fitted to inverse Fourier of frequency data, and vice versa
- error unstructured and indistinguishable from random

texture segmentation



- sample image on spatial uniform cartesian grid
- filter each spatial cell at different frequencies and orientations

“textons”

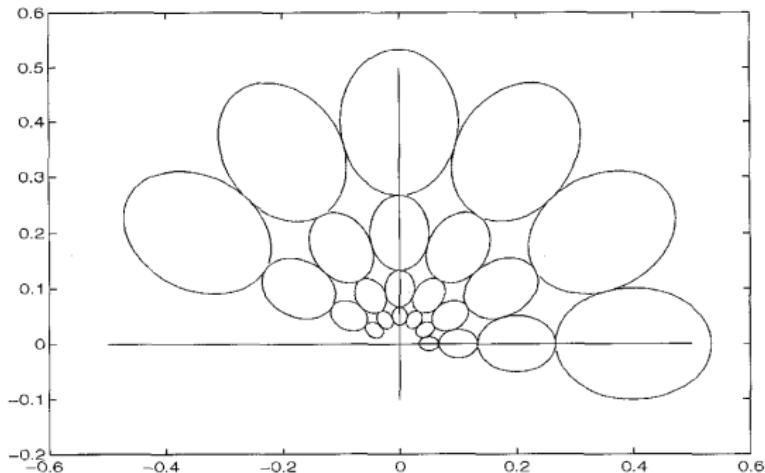


- see filter bank as frequency sampling on log-polar grid
- cluster filter (vector) responses into “textons”
- apply to iris recognition

visual descriptors

texture descriptors

[Manjunath and Ma 1996]



- same frequency sampling scheme
- filtering and global pooling in space domain
- popularized as part of MPEG-7 standard

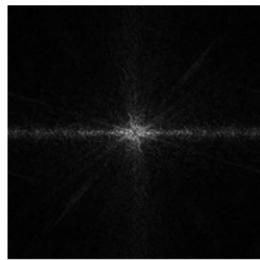
global descriptors



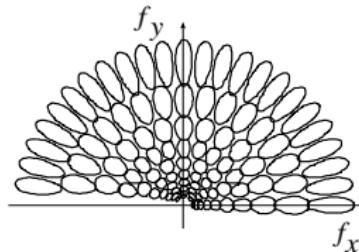
image



pre-processing



power spectrum



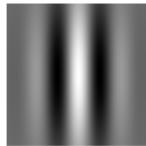
filter bank

- sampling scheme adapted to power spectrum statistics
- filtering and global pooling in frequency domain

sampling the frequency plane



frequency



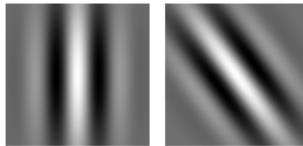
space

- space (\mathbf{x}) and frequency (\mathbf{u}) rotate together by θ
- scaling envelope (A) and carrier (\mathbf{u}_0) together
- 4d representation: position, scale, orientation

sampling the frequency plane



frequency



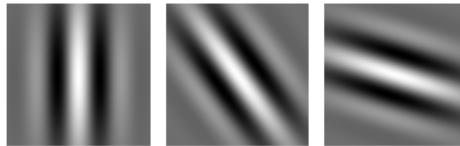
space

- space (\mathbf{x}) and frequency (\mathbf{u}) rotate together by θ
- scaling envelope (A) and carrier (\mathbf{u}_0) together
- 4d representation: position, scale, orientation

sampling the frequency plane



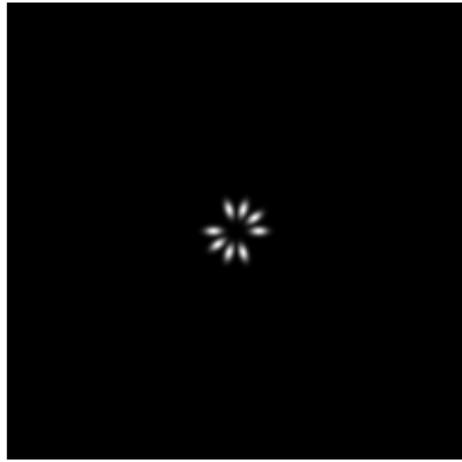
frequency



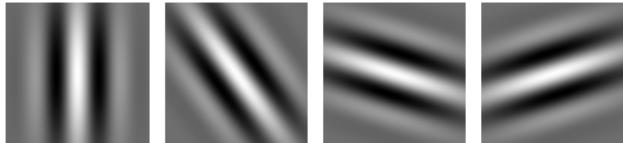
space

- space (\mathbf{x}) and frequency (\mathbf{u}) rotate together by θ
- scaling envelope (A) and carrier (\mathbf{u}_0) together
- 4d representation: position, scale, orientation

sampling the frequency plane



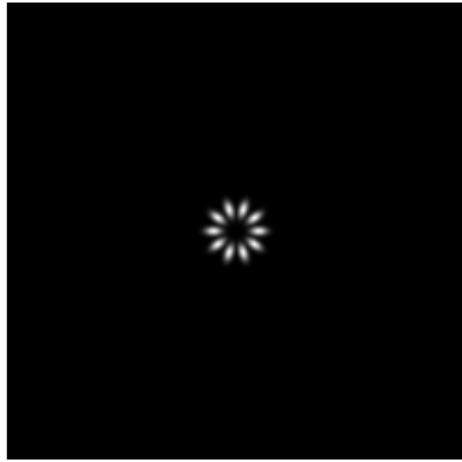
frequency



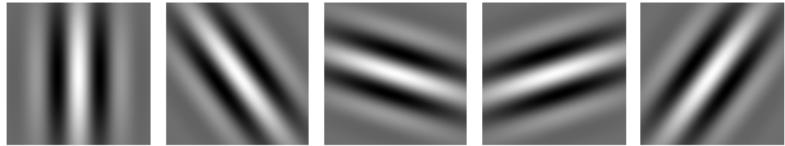
space

- space (\mathbf{x}) and frequency (\mathbf{u}) rotate together by θ
- scaling envelope (A) and carrier (\mathbf{u}_0) together
- 4d representation: position, scale, orientation

sampling the frequency plane



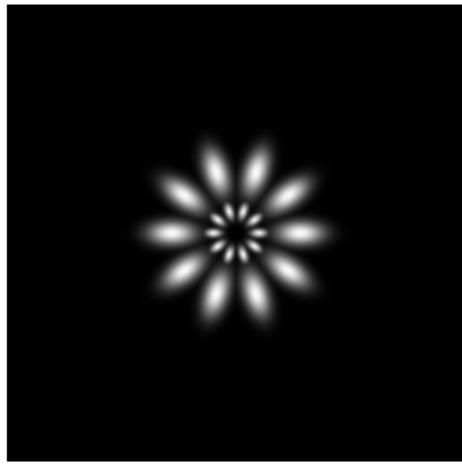
frequency



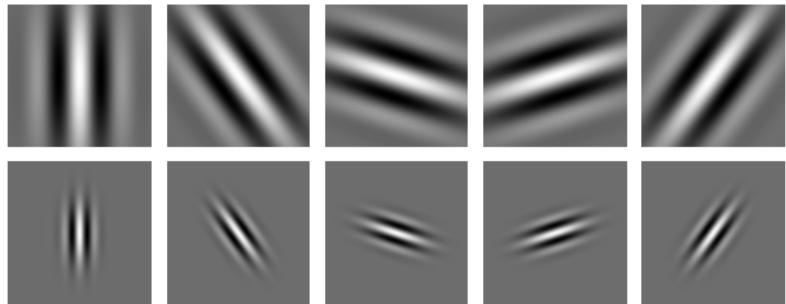
space

- space (\mathbf{x}) and frequency (\mathbf{u}) rotate together by θ
- scaling envelope (A) and carrier (\mathbf{u}_0) together
- 4d representation: position, scale, orientation

sampling the frequency plane



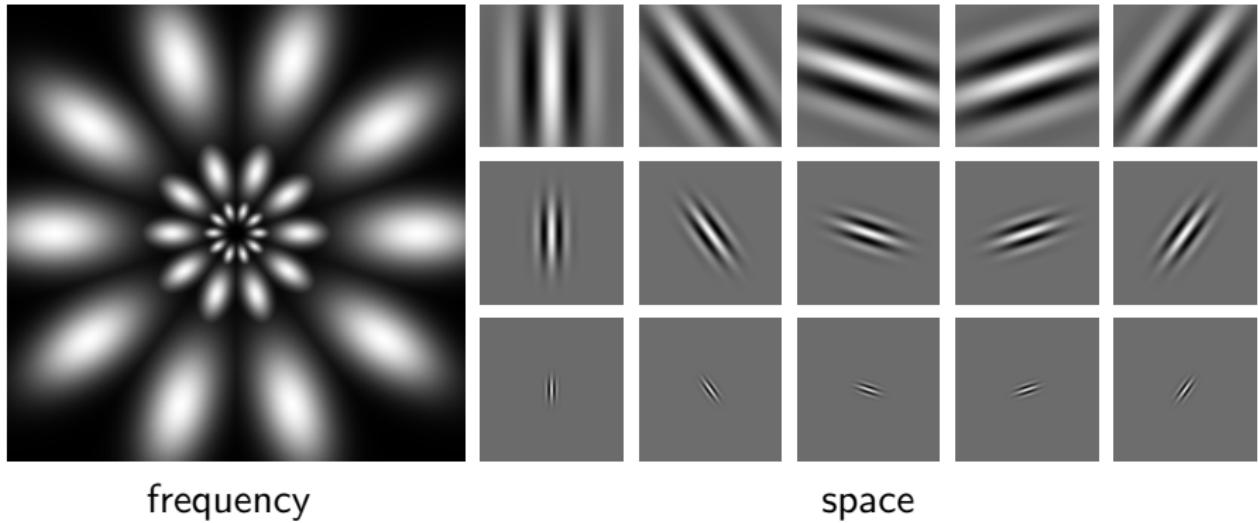
frequency



space

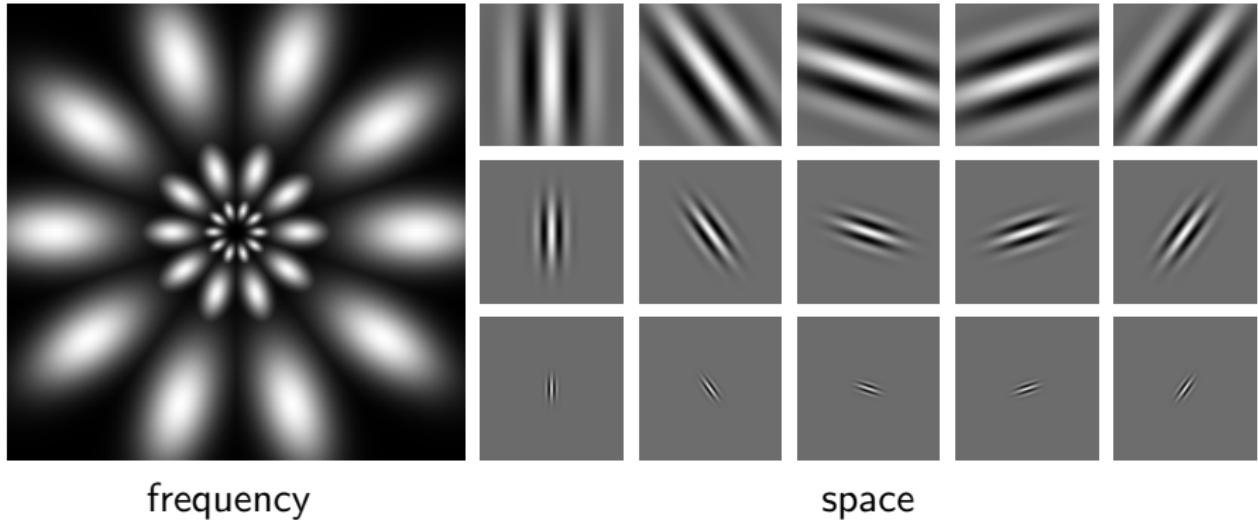
- space (x) and frequency (\mathbf{u}) rotate together by θ
- scaling envelope (A) and carrier (\mathbf{u}_0) together
- 4d representation: position, scale, orientation

sampling the frequency plane



- space (x) and frequency (\mathbf{u}) rotate together by θ
- scaling envelope (A) and carrier (\mathbf{u}_0) together
- 4d representation: position, scale, orientation

sampling the frequency plane



- space (x) and frequency (\mathbf{u}) rotate together by θ
- scaling envelope (A) and carrier (\mathbf{u}_0) together
- 4d representation: position, scale, orientation

from images to vectors

- suppose an image $f(\mathbf{x})$ is represented in frequency by $|F(\mathbf{u})|^2$
- suppose a template $h(\mathbf{x})$ (another image or an attribute) is also represented in frequency by

$$|H(\mathbf{u})|^2 = \sum_{n=1}^N h_n |G_n(\mathbf{u})|^2$$

where $\{G_n\}$ is a Gabor filter bank; let $\mathbf{h} = [h_1, \dots, h_N]$

- now define the vector $\mathbf{f} = [f_1, \dots, f_N]$ with

$$f_n = \int |F(\mathbf{u})|^2 |G_n(\mathbf{u})|^2 d\mathbf{u}$$

- and measure the similarity of f, h by the inner product

$$\int |F(\mathbf{u})|^2 |H(\mathbf{u})|^2 d\mathbf{u} = \sum_{n=1}^N f_n h_n = \langle \mathbf{f}, \mathbf{h} \rangle$$

from images to vectors

- suppose an image $f(\mathbf{x})$ is represented in frequency by $|F(\mathbf{u})|^2$
- suppose a template $h(\mathbf{x})$ (another image or an attribute) is also represented in frequency by

$$|H(\mathbf{u})|^2 = \sum_{n=1}^N h_n |G_n(\mathbf{u})|^2$$

where $\{G_n\}$ is a Gabor filter bank; let $\mathbf{h} = [h_1, \dots, h_N]$

- now define the vector $\mathbf{f} = [f_1, \dots, f_N]$ with

$$f_n = \int |F(\mathbf{u})|^2 |G_n(\mathbf{u})|^2 d\mathbf{u}$$

- and measure the similarity of f, h by the inner product

$$\int |F(\mathbf{u})|^2 |H(\mathbf{u})|^2 d\mathbf{u} = \sum_{n=1}^N f_n h_n = \langle \mathbf{f}, \mathbf{h} \rangle$$

from images to vectors

- suppose an image $f(\mathbf{x})$ is represented in frequency by $|F(\mathbf{u})|^2$
- suppose a template $h(\mathbf{x})$ (another image or an attribute) is also represented in frequency by

$$|H(\mathbf{u})|^2 = \sum_{n=1}^N h_n |G_n(\mathbf{u})|^2$$

where $\{G_n\}$ is a Gabor filter bank; let $\mathbf{h} = [h_1, \dots, h_N]$

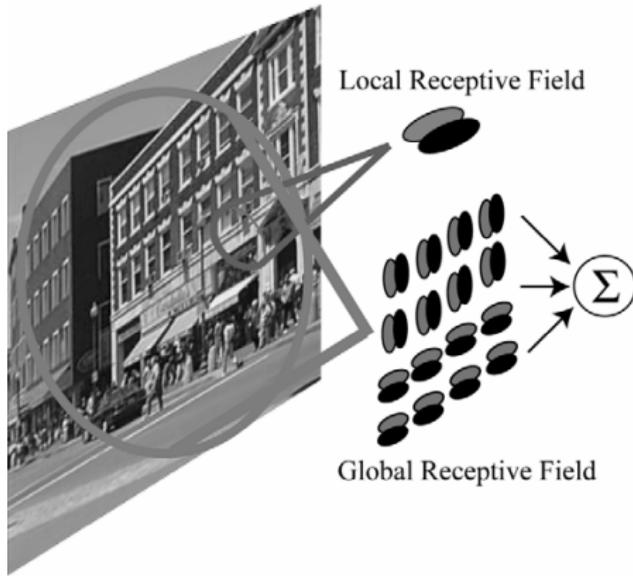
- now define the vector $\mathbf{f} = [f_1, \dots, f_N]$ with

$$f_n = \int |F(\mathbf{u})|^2 |G_n(\mathbf{u})|^2 d\mathbf{u}$$

- and measure the similarity of f, h by the inner product

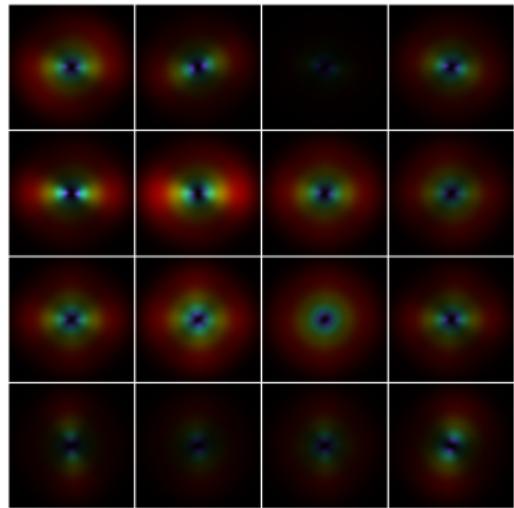
$$\int |F(\mathbf{u})|^2 |H(\mathbf{u})|^2 d\mathbf{u} = \sum_{n=1}^N f_n h_n = \langle \mathbf{f}, \mathbf{h} \rangle$$

global vs. local receptive fields



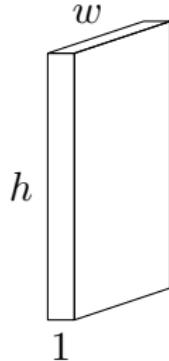
- pool filter responses only locally
 - next level in hierarchy can apply different spatial weights

the gist descriptor



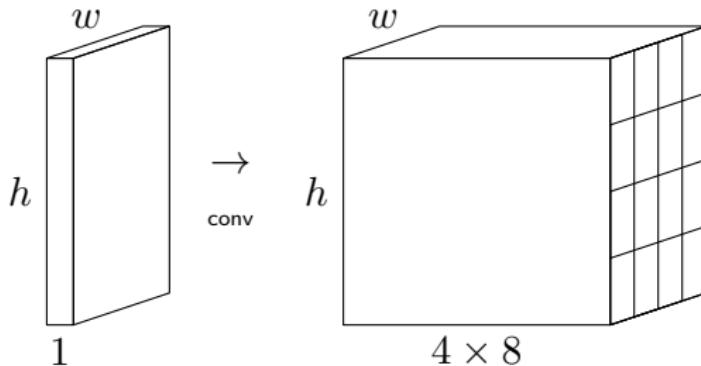
- apply filter bank to entire image in frequency domain
- partition image in 4×4 cells
- average pooling of filter responses per cell

gist pipeline



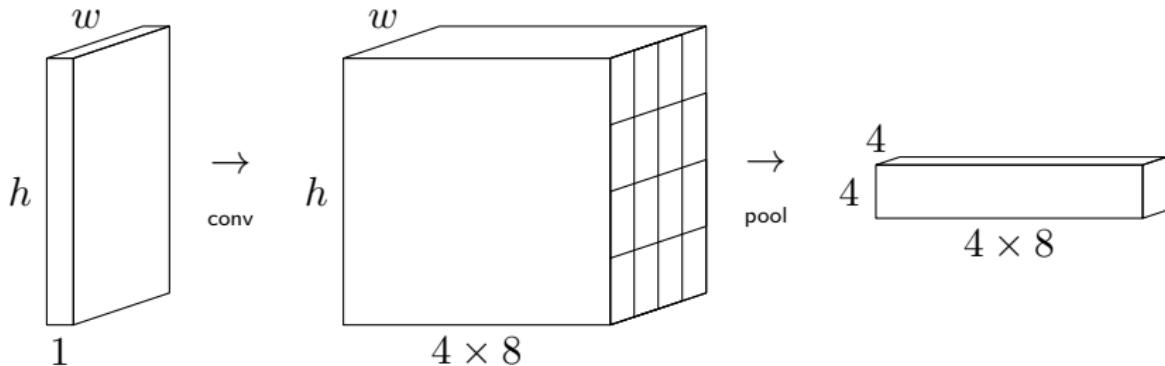
- 3-channel RGB input → 1-channel gray-scale
- apply filters at $4 \text{ scales} \times 8 \text{ orientations}$
- average pooling on 4×4 cells → descriptor of length 512

gist pipeline



- 3-channel RGB input → 1-channel gray-scale
 - apply filters at 4 scales \times 8 orientations
 - average pooling on 4×4 cells → descriptor of length 512

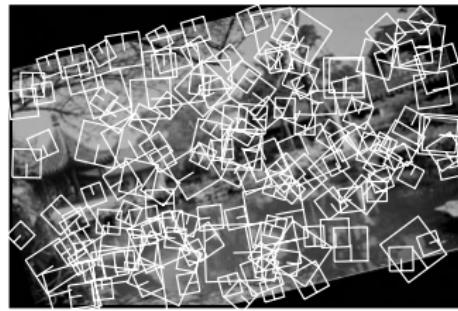
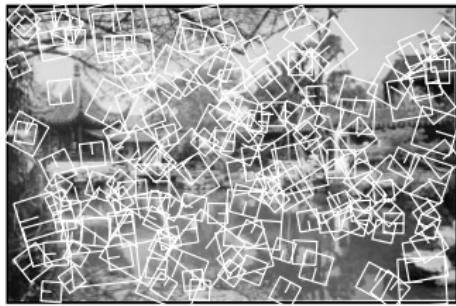
gist pipeline



- 3-channel RGB input → 1-channel gray-scale
 - apply filters at 4 scales \times 8 orientations
 - average pooling on 4×4 cells → descriptor of length 512

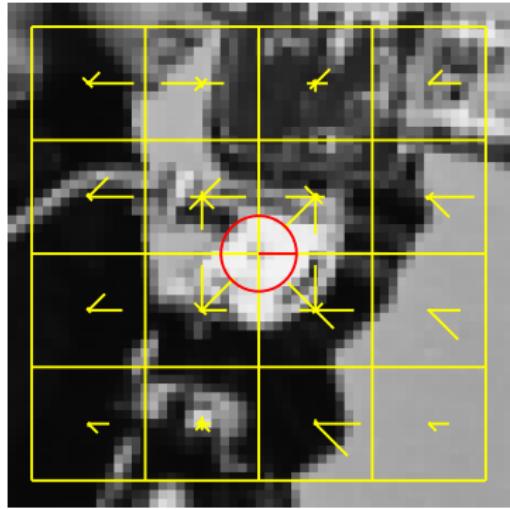
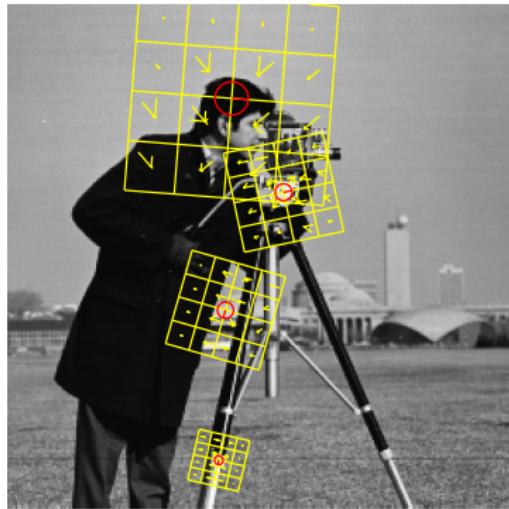
scale-invariant feature transform

[Lowe 1999]



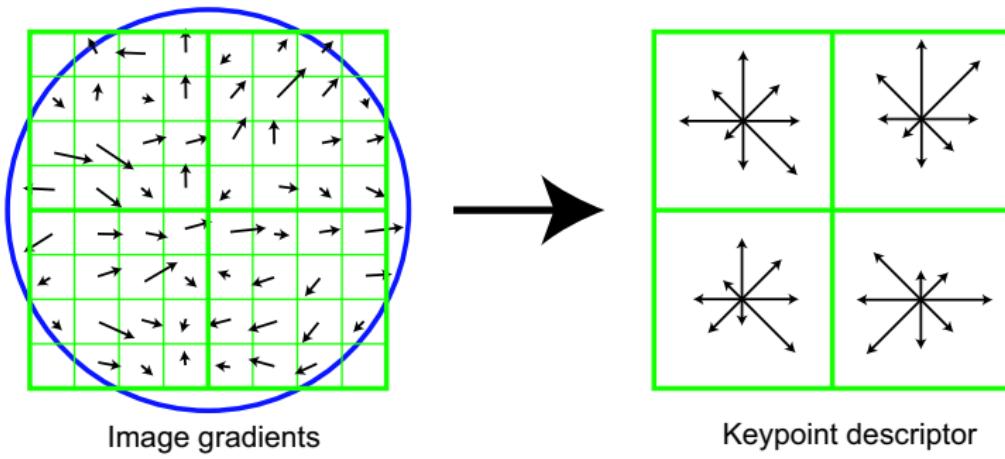
- detect a sparse set of “stable” features (rectangular patches), **equivariant** to translation, scale and rotation

scale-invariant feature transform



- for each patch
 - normalize with respect to scale and orientation
 - construct a histogram of gradient orientations

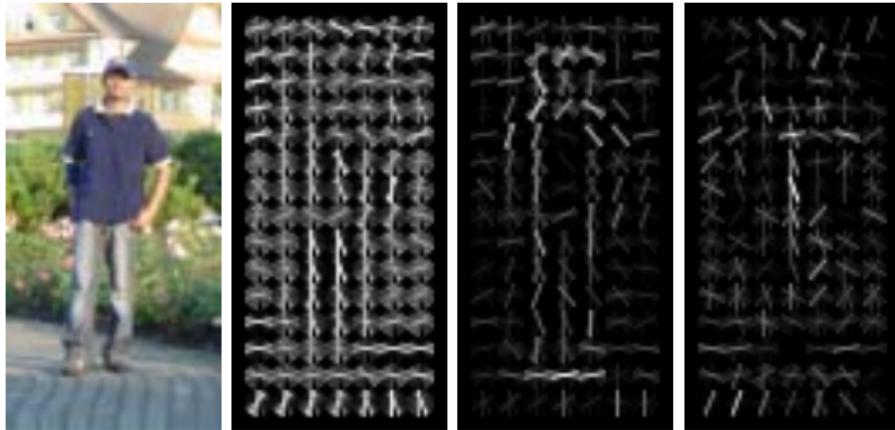
the SIFT descriptor



- votes in 8-bin orientation histograms weighted by magnitude and by Gaussian window on patch
 - histograms pooled over 4×4 cells, trilinear interpolation
 - 128-dimensional descriptor, normalized, clipped at 0.2, normalized

histogram of oriented gradients

[Dalal and Triggs 2005]



- applied to person detection by sliding window and SVM
- classifier learns positive and negative weights on positions and orientations
- switch focus back to dense features for classification

the HOG descriptor



- applied densely to adjacent cells of 8×8 pixels
- no scale or orientation normalization; just single-scale
- normalized by overlapping blocks of 3×3 cells—redundant

so what is a histogram?

- consider a histogram h over integers $C = \{0, 1, 2, 3, 4\}$, computed from the following samples:

$$\begin{array}{rcl} C & = & \{ 0 \ 1 \ 2 \ 3 \ 4 \ } \\ \hline 3 & \rightarrow & (0 \ 0 \ 0 \ 1 \ 0) \\ 2 & \rightarrow & (0 \ 0 \ 1 \ 0 \ 0) \\ 0 & \rightarrow & (1 \ 0 \ 0 \ 0 \ 0) \\ 3 & \rightarrow & (0 \ 0 \ 0 \ 1 \ 0) \\ 2 & \rightarrow & (0 \ 0 \ 1 \ 0 \ 0) \\ 2 & \rightarrow & (0 \ 0 \ 1 \ 0 \ 0) \\ \hline h & = & (1 \ 0 \ 3 \ 2 \ 0) \ / \ 6 \end{array}$$

- each sample is **encoded** (*hard-assigned*) into a vector in \mathbb{R}^5 ; all such vectors are **pooled** (*averaged*) into one vector $h \in \mathbb{R}^5$
- encoding is always **nonlinear** and pooling is **orderless**
- C is a **codebook** or **vocabulary**

so what is a histogram?

- consider a histogram h over integers $C = \{0, 1, 2, 3, 4\}$, computed from the following samples:

$$\begin{array}{rcl} C & = & \{ 0 \ 1 \ 2 \ 3 \ 4 \ } \\ \hline 3 & \rightarrow & (0 \ 0 \ 0 \ 1 \ 0) \\ 2 & \rightarrow & (0 \ 0 \ 1 \ 0 \ 0) \\ 0 & \rightarrow & (1 \ 0 \ 0 \ 0 \ 0) \\ 3 & \rightarrow & (0 \ 0 \ 0 \ 1 \ 0) \\ 2 & \rightarrow & (0 \ 0 \ 1 \ 0 \ 0) \\ 2 & \rightarrow & (0 \ 0 \ 1 \ 0 \ 0) \\ \hline h & = & (1 \ 0 \ 3 \ 2 \ 0) \ / \ 6 \end{array}$$

- each sample is **encoded** (*hard-assigned*) into a vector in \mathbb{R}^5 ; all such vectors are **pooled** (*averaged*) into one vector $h \in \mathbb{R}^5$
- encoding is always **nonlinear** and pooling is **orderless**
- C is a **codebook** or **vocabulary**

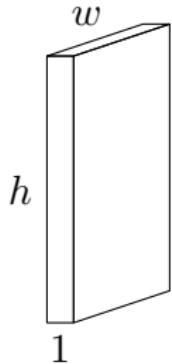
so what is a histogram?

- consider a histogram h over integers $C = \{0, 1, 2, 3, 4\}$, computed from the following samples:

$$\begin{array}{rcl} C & = & \{ 0 \ 1 \ 2 \ 3 \ 4 \ } \\ \hline 3 & \rightarrow & (0 \ 0 \ 0 \ 1 \ 0) \\ 2 & \rightarrow & (0 \ 0 \ 1 \ 0 \ 0) \\ 0 & \rightarrow & (1 \ 0 \ 0 \ 0 \ 0) \\ 3 & \rightarrow & (0 \ 0 \ 0 \ 1 \ 0) \\ 2 & \rightarrow & (0 \ 0 \ 1 \ 0 \ 0) \\ 2 & \rightarrow & (0 \ 0 \ 1 \ 0 \ 0) \\ \hline h & = & (1 \ 0 \ 3 \ 2 \ 0) \ / \ 6 \end{array}$$

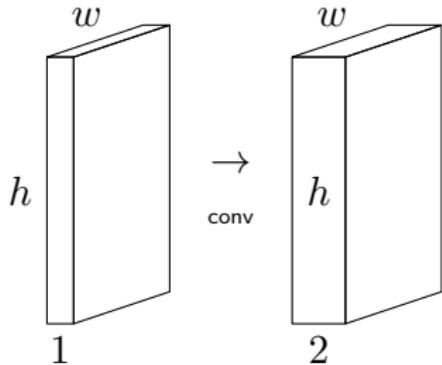
- each sample is **encoded** (*hard-assigned*) into a vector in \mathbb{R}^5 ; all such vectors are **pooled** (*averaged*) into one vector $h \in \mathbb{R}^5$
- encoding is always **nonlinear** and pooling is **orderless**
- C is a **codebook** or **vocabulary**

SIFT (HOG) pipeline



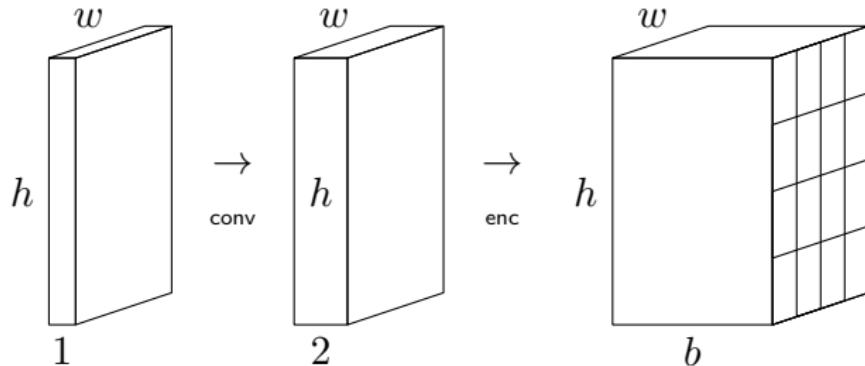
- 3-channel patch (**image**) RGB input → 1-channel gray-scale
 - compute gradient magnitude & orientation
 - encode into $b = 8$ (**9**) orientation bins
 - average pooling on $c = 4 \times 4$ ($\lfloor w/8 \rfloor \times \lfloor h/8 \rfloor$) cells
 - descriptor of length $c \times b = 128$ (**block-normalize** → $c \times (3 \times 3) \times b$)

SIFT (HOG) pipeline



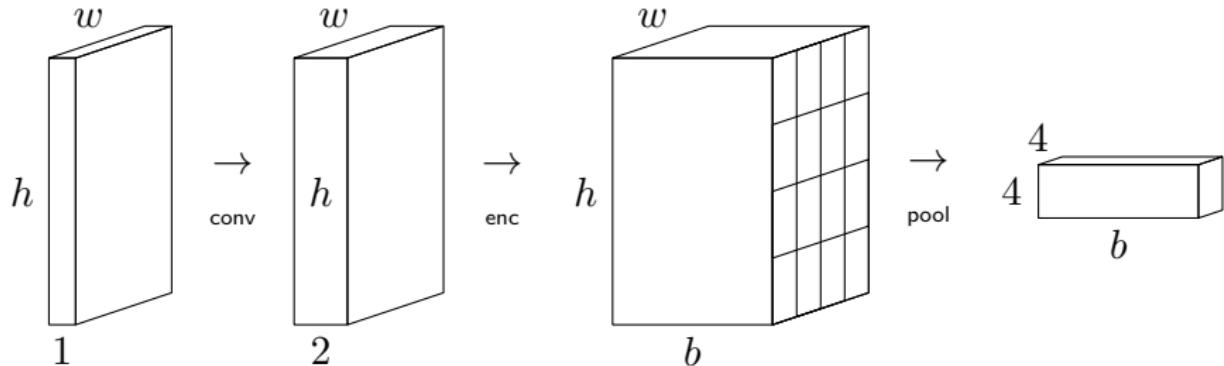
- 3-channel patch (**image**) RGB input → 1-channel gray-scale
- compute gradient magnitude & orientation
- encode into $b = 8$ (**9**) orientation bins
- average pooling on $c = 4 \times 4$ ($\lfloor w/8 \rfloor \times \lfloor h/8 \rfloor$) cells
- descriptor of length $c \times b = 128$ (**block-normalize** → $c \times (3 \times 3) \times b$)

SIFT (HOG) pipeline



- 3-channel patch (**image**) RGB input \rightarrow 1-channel gray-scale
- compute gradient magnitude & orientation
- encode into $b = 8$ (**9**) orientation bins
 - average pooling on $c = 4 \times 4$ ($\lfloor w/8 \rfloor \times \lfloor h/8 \rfloor$) cells
 - descriptor of length $c \times b = 128$ (**block-normalize** $\rightarrow c \times (3 \times 3) \times b$)

SIFT (HOG) pipeline



- 3-channel patch (**image**) RGB input \rightarrow 1-channel gray-scale
- compute gradient magnitude & orientation
- encode into $b = 8$ (**9**) orientation bins
- average pooling on $c = 4 \times 4$ ($\lfloor w/8 \rfloor \times \lfloor h/8 \rfloor$) cells
- descriptor of length $c \times b = 128$ (**block-normalize** $\rightarrow c \times (3 \times 3) \times b$)

embeddings

back to Gabor

- let us use the following edge pattern



- rotate it by all $\theta \in [0, 2\pi]$
- for each θ , filter (take dot product) with a bank of antisymmetric Gabor filters at 5 orientations, single scale
- turns out, the filter bank provides an encoding of θ in \mathbb{R}^5 : soft assignment
- then, spatial pooling gives nothing but an orientation histogram

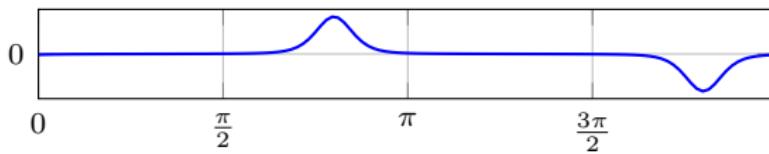
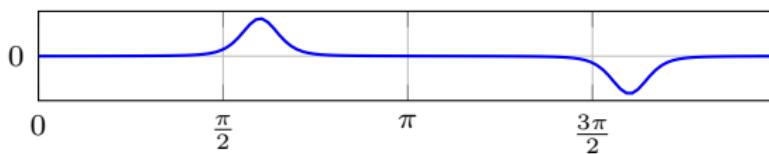
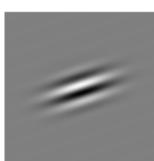
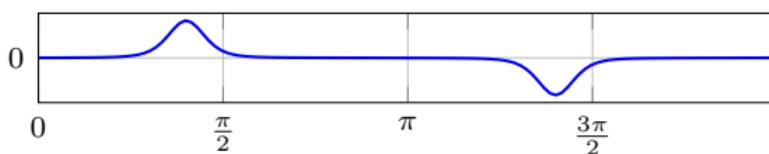
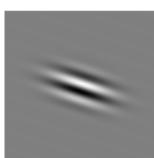
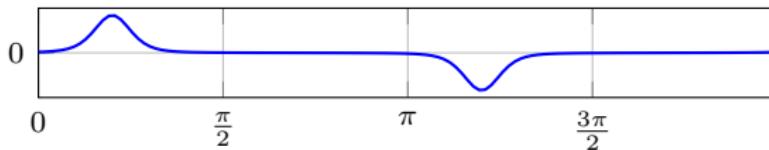
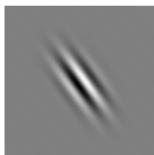
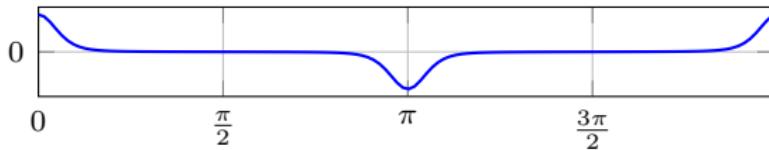
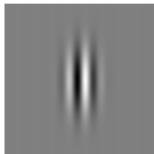
back to Gabor

- let us use the following edge pattern

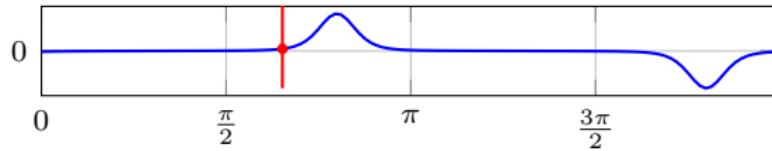
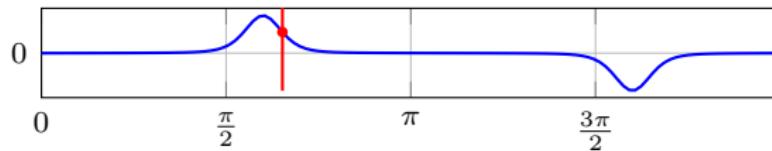
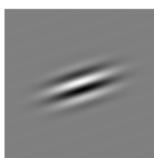
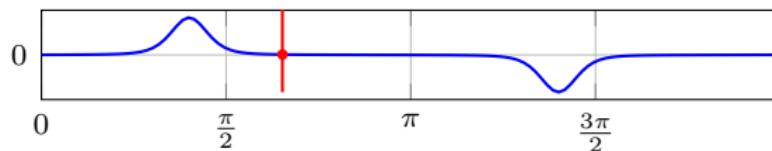
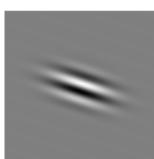
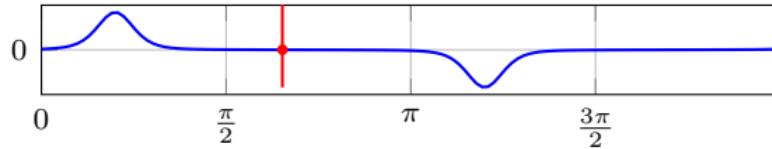
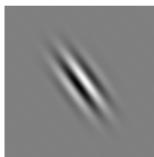
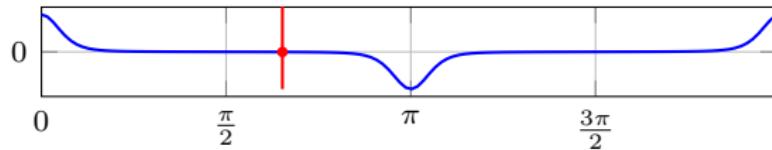
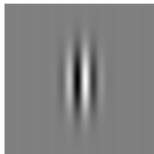


- rotate it by all $\theta \in [0, 2\pi]$
- for each θ , filter (take dot product) with a bank of antisymmetric Gabor filters at 5 orientations, single scale
- turns out, the filter bank provides an encoding of θ in \mathbb{R}^5 : soft assignment
- then, spatial pooling gives nothing but an orientation histogram

back to Gabor



back to Gabor

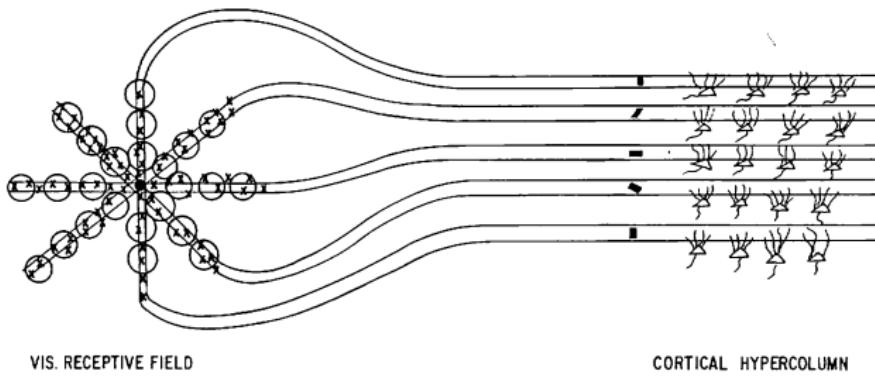


nonlinear mappings

- Q: we said convolution is linear; now, once we have a gradient orientation measurement, why do we need a nonlinear function?

nonlinear mappings

- Q: we said convolution is linear; now, once we have a gradient orientation measurement, why do we need a nonlinear function?
 - convolution is linear in the image; but if the image is rotated by θ , itself is a nonlinear function of θ
 - what we are doing is, mapping to another space where scaling and rotation of the image behave like translation



on manifolds

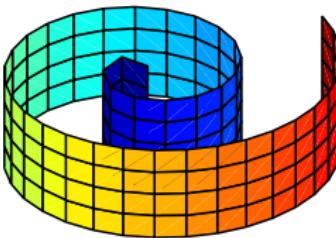
- an image of resolution 320×200 is a vector in $\mathcal{I} = \mathbb{R}^{64,000}$; are all such vectors equally likely?
- an object seen at different scales and orientations only spans a 2-dimensional smooth manifold in \mathcal{I}

and we would like to express scale and orientation as two natural coordinates

- how would we go about expressing perspective transformation? attributes of handwritten characters? poses of a human body? occluded surfaces? species of dogs?

on manifolds

- an image of resolution 320×200 is a vector in $\mathcal{I} = \mathbb{R}^{64,000}$; are all such vectors equally likely?
- an object seen at different scales and orientations only spans a 2-dimensional smooth manifold in \mathcal{I}

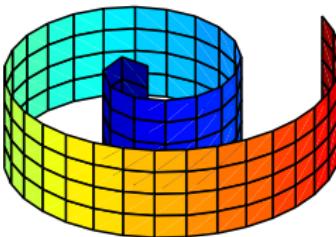


and we would like to express scale and orientation as two natural coordinates

- how would we go about expressing perspective transformation? attributes of handwritten characters? poses of a human body? occluded surfaces? species of dogs?

on manifolds

- an image of resolution 320×200 is a vector in $\mathcal{I} = \mathbb{R}^{64,000}$; are all such vectors equally likely?
- an object seen at different scales and orientations only spans a 2-dimensional smooth manifold in \mathcal{I}



and we would like to express scale and orientation as two natural coordinates

- how would we go about expressing perspective transformation? attributes of handwritten characters? poses of a human body? occluded surfaces? species of dogs?

hierarchy

- at each level, nonlinearly encode each local (e.g. pixel) representation according to a codebook, followed by pooling
- scale and orientation are just two dimensions; a codebook is just a dense grid
- a 3-scale, 6-orientation filter response is 18-dimensional; a dense grid is not an option
- learn the codebook from data!

hierarchy

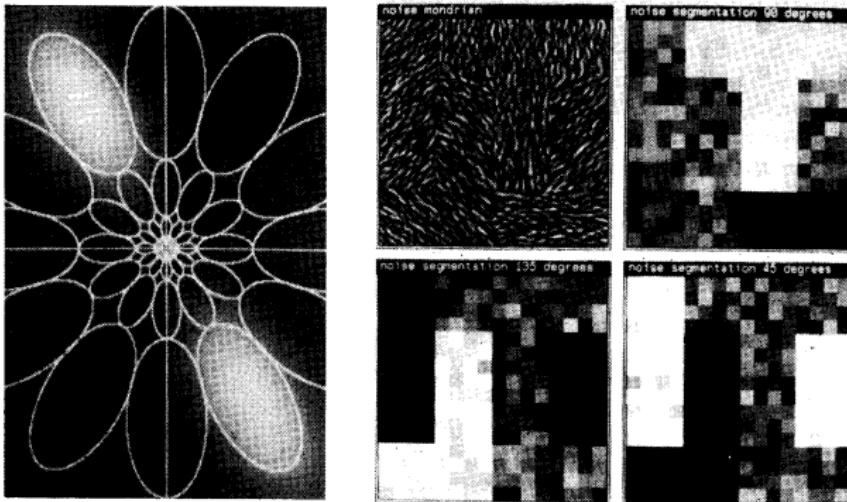
- at each level, nonlinearly encode each local (e.g. pixel) representation according to a codebook, followed by pooling
- scale and orientation are just two dimensions; a codebook is just a dense grid
- a 3-scale, 6-orientation filter response is 18-dimensional; a dense grid is not an option
- learn the codebook from data!

hierarchy

- at each level, nonlinearly encode each local (e.g. pixel) representation according to a codebook, followed by pooling
- scale and orientation are just two dimensions; a codebook is just a dense grid
- a 3-scale, 6-orientation filter response is 18-dimensional; a dense grid is not an option
- learn the codebook from data!

back to textons

[Daugman 1988]



- see filter bank as frequency sampling on log-polar grid
- cluster 3×6 filter (vector) responses into “textons”
- apply to iris recognition

textons

[Malik et al. 1999]



oriented filter bank



image



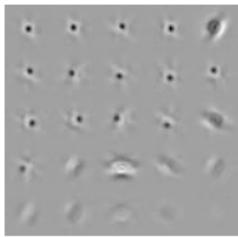
texture segmentation

- textons (re-)defined as clusters of filter responses
- regions described by texton histograms

textons



image



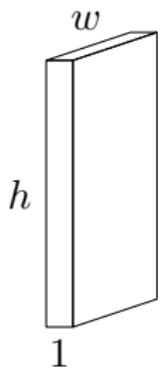
textons



channels

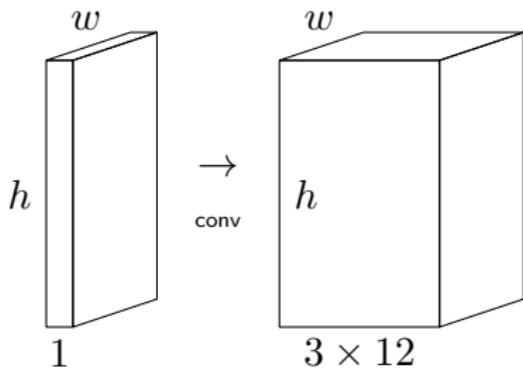
- each pixel mapped to a filter response vector of length 3×12
- vectors clustered by k -means into $k = 25$ “texton” centroids
- each pixel assigned to a texton

texton pipeline



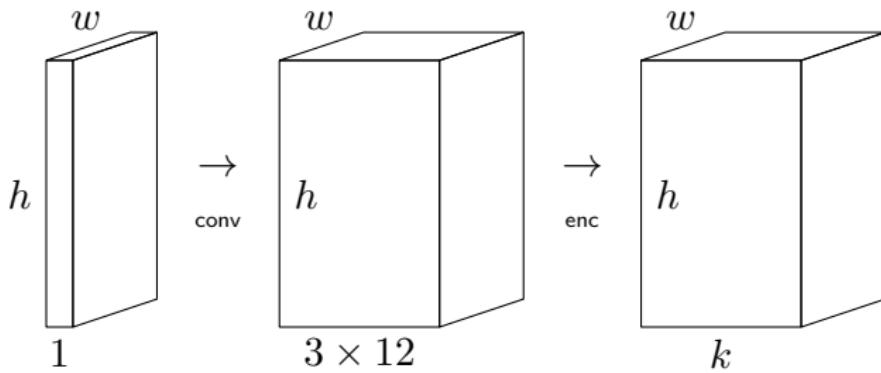
- 3-channel RGB input → 1-channel gray-scale
 - apply filters at 3 scales × 12 orientations
 - point-wise encoding (hard assignment) on $k = 25$ textons
 - stride-1 average pooling on overlapping neighborhoods

texton pipeline



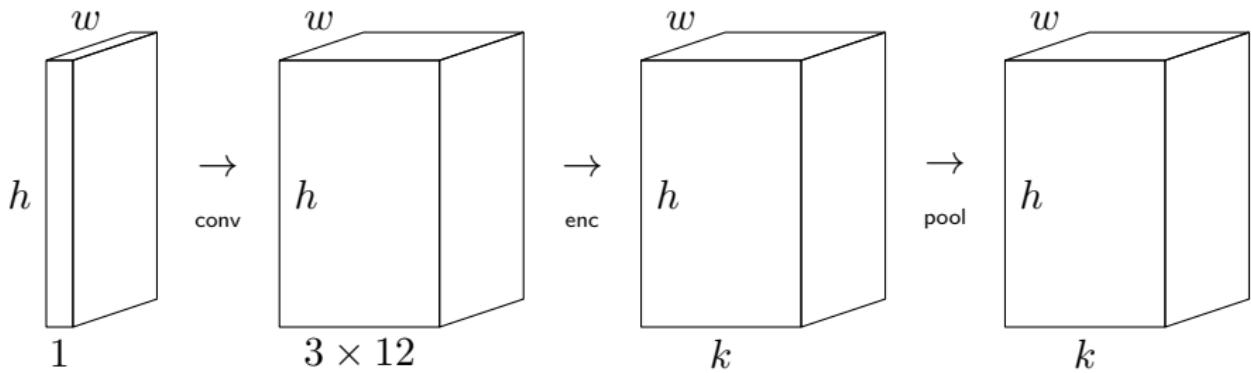
- 3-channel RGB input \rightarrow 1-channel gray-scale
- apply filters at 3 scales \times 12 orientations
- point-wise encoding (hard assignment) on $k = 25$ textons
- stride-1 average pooling on overlapping neighborhoods

texton pipeline



- 3-channel RGB input \rightarrow 1-channel gray-scale
- apply filters at 3 scales \times 12 orientations
- point-wise encoding (hard assignment) on $k = 25$ textons
- stride-1 average pooling on overlapping neighborhoods

texton pipeline



- 3-channel RGB input \rightarrow 1-channel gray-scale
- apply filters at $3 \text{ scales} \times 12 \text{ orientations}$
- point-wise encoding (hard assignment) on $k = 25$ textons
- stride-1 average pooling on overlapping neighborhoods

bag of words (BoW)

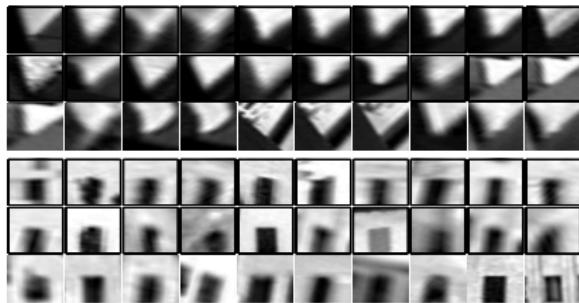
[Sivic and Zisserman 2003]



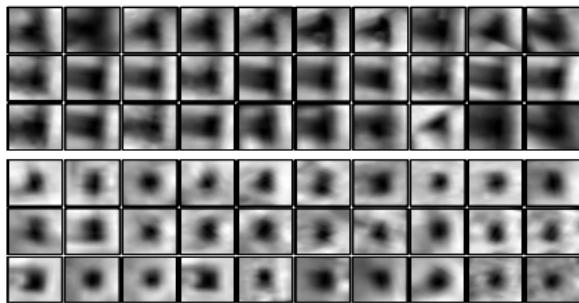
- two types of sparse features detected
- SIFT descriptors extracted from a dataset of video frames

bag of words: retrieval

[Sivic and Zisserman 2003]



Harris affine
6k words



maximally stable
10k words

- “visual words” defined as clusters of SIFT descriptors learned from the dataset
- images described by visual word histograms
- matching is reduced to sparse dot product → fast retrieval

bag of words: classification

[Csurka et al. 2004]



features



visual words



phones, books, cars



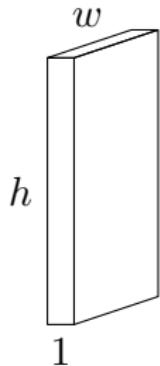
bikes, buildings, cars



buildings, cars, faces

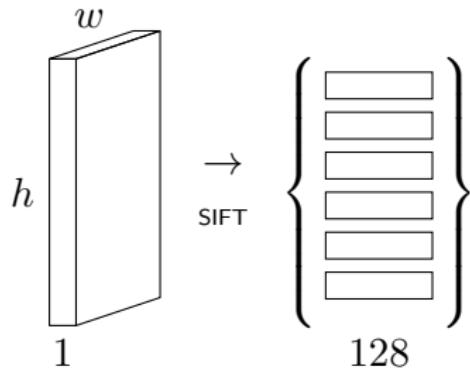
- same representation, $k = 1000$ words, naive bayes or SVM classifier
- features soon to be replaced dense multiscale HOG or SIFT

bag of words pipeline



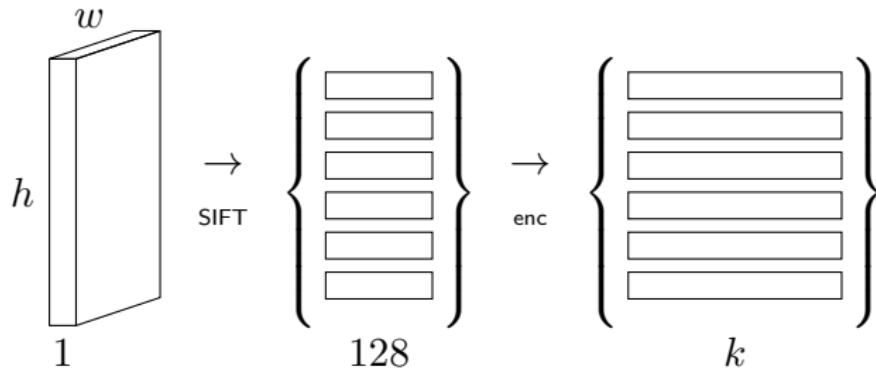
- 3-channel RGB input \rightarrow 1-channel gray-scale
- set of ~ 1000 features \times 128-dim SIFT descriptors
- element-wise encoding (hard assignment) on $k \sim 10^4$ visual words
- global sum pooling, ℓ^2 normalization

bag of words pipeline



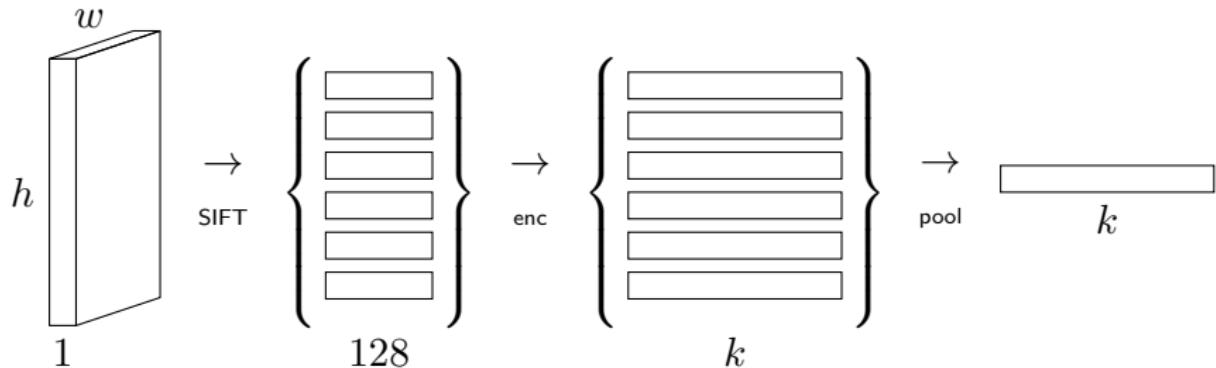
- 3-channel RGB input → 1-channel gray-scale
- set of ~ 1000 features \times 128-dim SIFT descriptors
- element-wise encoding (hard assignment) on $k \sim 10^4$ visual words
- global sum pooling, ℓ^2 normalization

bag of words pipeline



- 3-channel RGB input \rightarrow 1-channel gray-scale
- set of ~ 1000 features \times 128-dim SIFT descriptors
- element-wise encoding (hard assignment) on $k \sim 10^4$ visual words
- global sum pooling, ℓ^2 normalization

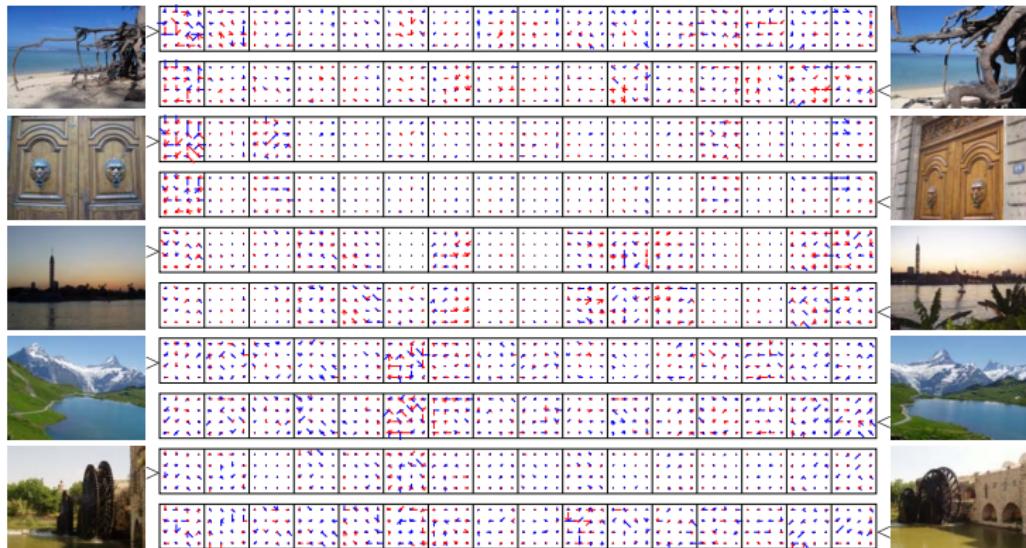
bag of words pipeline



- 3-channel RGB input \rightarrow 1-channel gray-scale
- set of ~ 1000 features \times 128-dim SIFT descriptors
- element-wise encoding (hard assignment) on $k \sim 10^4$ visual words
- global sum pooling, ℓ^2 normalization

vector of locally aggregated descriptors (VLAD)

[Jégou et al. 2010]



- encoding yields a vector per visual word, rather than a scalar frequency
 - this vector is 128-dimensional like SIFT descriptors

VLAD definition

- input vectors: $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$
- vector quantizer: $q : \mathbb{R}^d \rightarrow C \subset \mathbb{R}^d$, $C = \{c_1, \dots, c_k\}$

$$q(x) = \arg \min_{c \in C} \|x - c\|^2$$

- residual vector

$$r(x) = x - q(x)$$

- residual pooling per cell

$$V_c(X) = \sum_{\substack{x \in X \\ q(x)=c}} r(x) = \sum_{\substack{x \in X \\ q(x)=c}} x - q(x)$$

- VLAD vector (up to normalization)

$$\mathcal{V}(X) = [V_{c_1}(X), \dots, V_{c_k}(X)]$$

VLAD definition

- input vectors: $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$
- vector quantizer: $q : \mathbb{R}^d \rightarrow C \subset \mathbb{R}^d$, $C = \{c_1, \dots, c_k\}$

$$q(x) = \arg \min_{c \in C} \|x - c\|^2$$

- residual vector

$$r(x) = x - q(x)$$

- residual pooling per cell

$$V_c(X) = \sum_{\substack{x \in X \\ q(x)=c}} r(x) = \sum_{\substack{x \in X \\ q(x)=c}} x - q(x)$$

- VLAD vector (up to normalization)

$$\mathcal{V}(X) = [V_{c_1}(X), \dots, V_{c_k}(X)]$$

VLAD definition

- input vectors: $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$
- vector quantizer: $q : \mathbb{R}^d \rightarrow C \subset \mathbb{R}^d$, $C = \{c_1, \dots, c_k\}$

$$q(x) = \arg \min_{c \in C} \|x - c\|^2$$

- residual vector

$$r(x) = x - q(x)$$

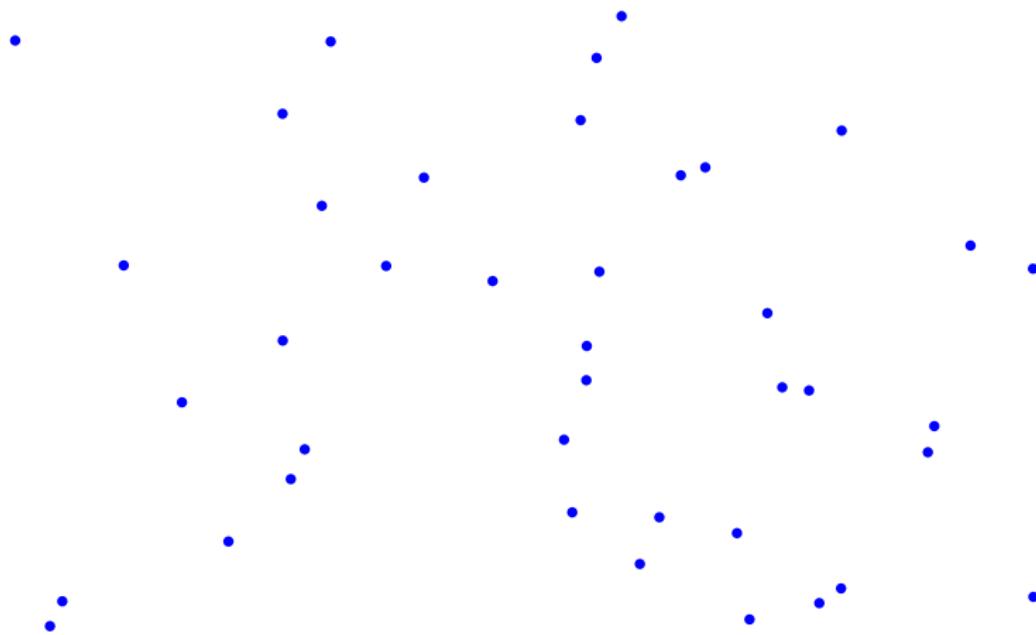
- residual pooling per cell

$$V_c(X) = \sum_{\substack{x \in X \\ q(x)=c}} r(x) = \sum_{\substack{x \in X \\ q(x)=c}} x - q(x)$$

- VLAD vector (up to normalization)

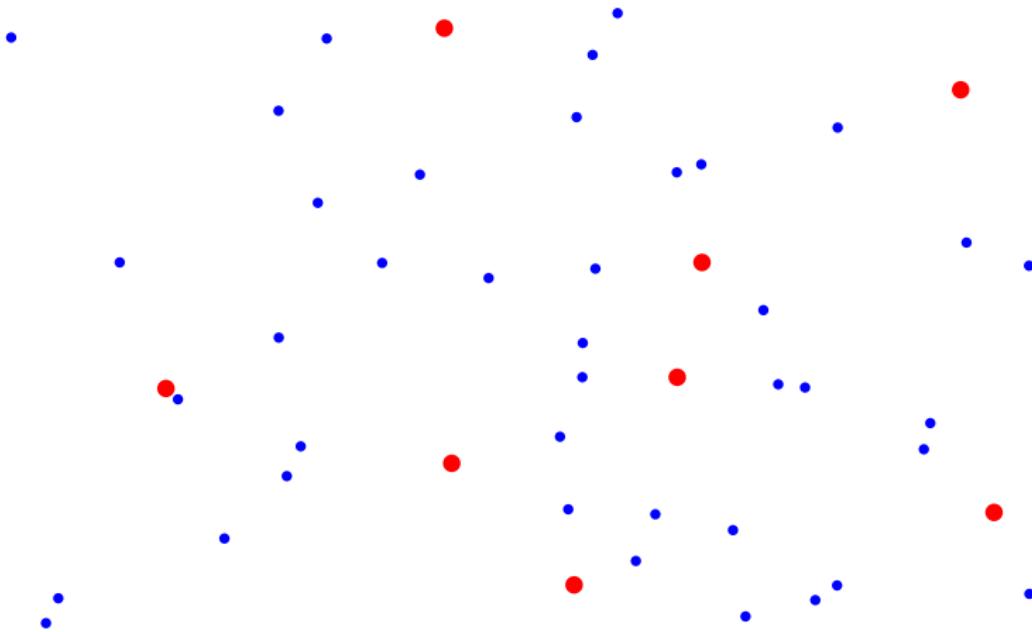
$$\mathcal{V}(X) = [V_{c_1}(X), \dots, V_{c_k}(X)]$$

VLAD geometry



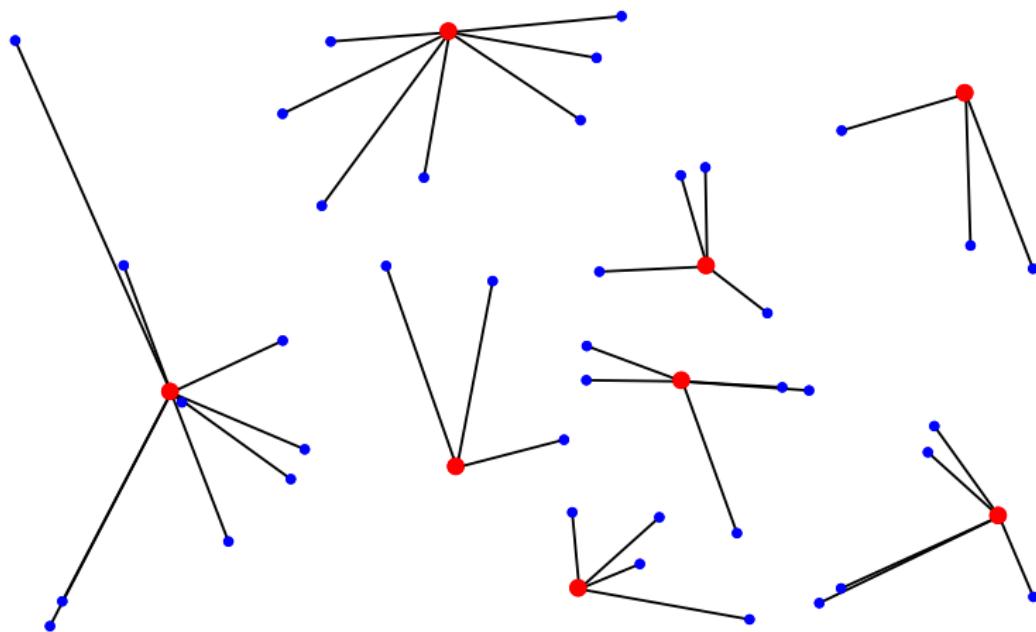
- **input vectors** – codebook – residuals – pooling

VLAD geometry



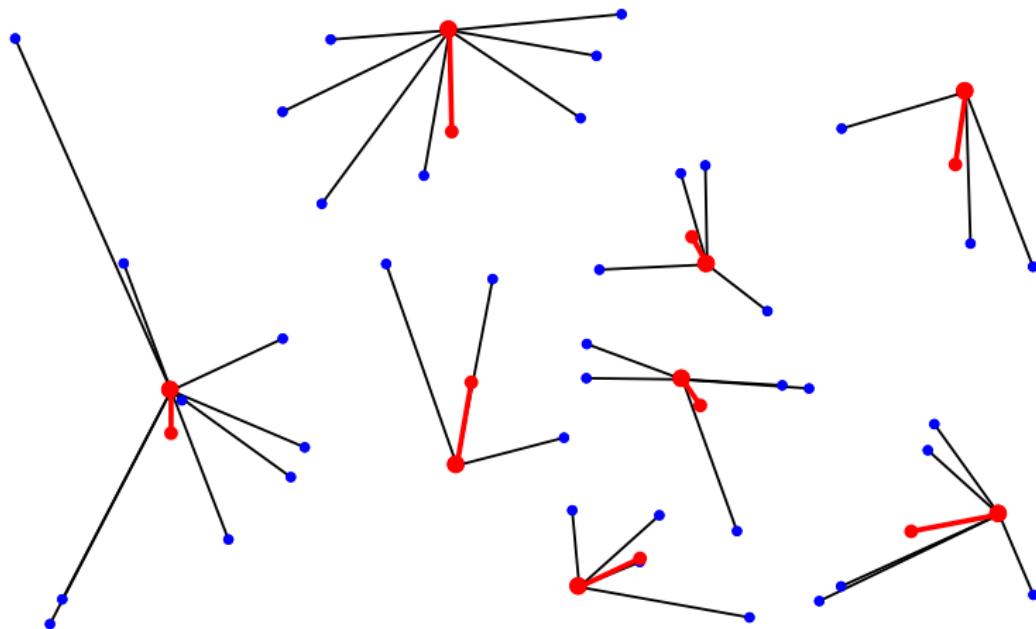
- input vectors – codebook – residuals – pooling

VLAD geometry



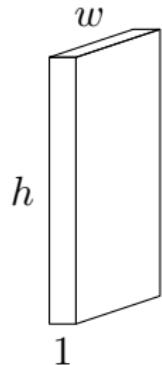
- input vectors – codebook – residuals – pooling

VLAD geometry



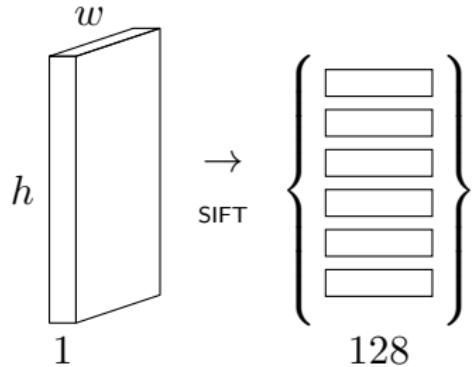
- input vectors – codebook – residuals – pooling

VLAD pipeline



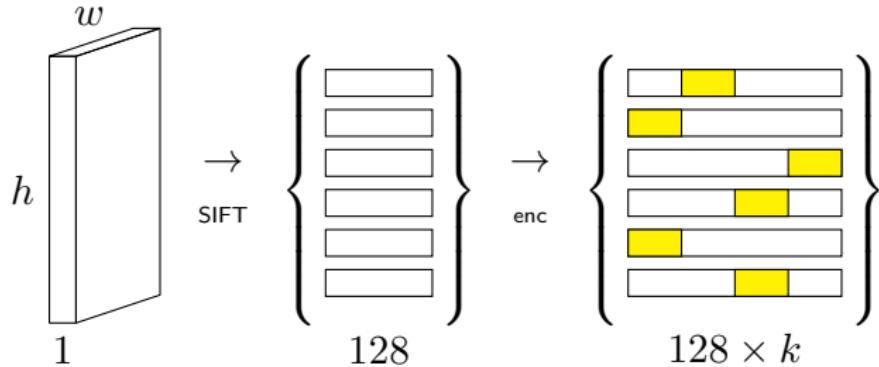
- 3-channel RGB input → 1-channel gray-scale
- set of ~ 1000 features \times 128-dim SIFT descriptors
- element-wise encoding (hard assignment) on $k \sim 16$ visual words
- encoding now yields a residual vector rather than a scalar vote
- global sum pooling, ℓ^2 normalization

VLAD pipeline



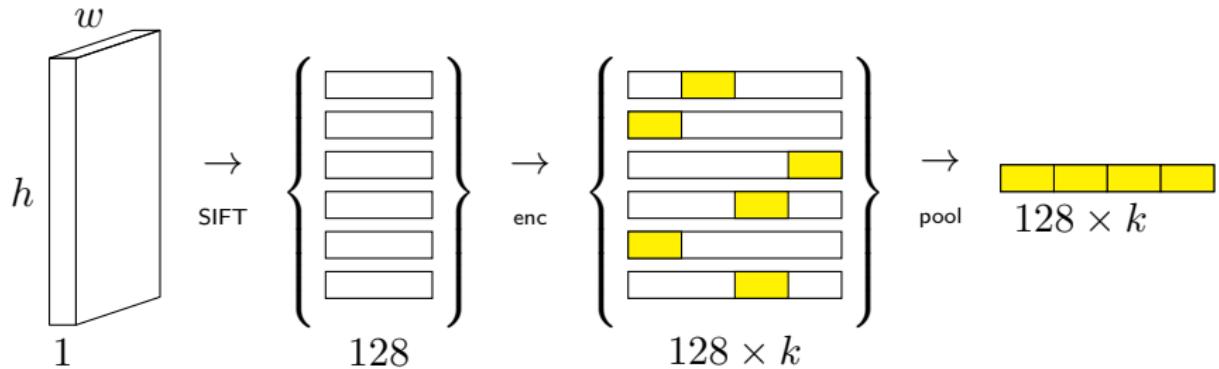
- 3-channel RGB input → 1-channel gray-scale
- set of ~ 1000 features \times 128-dim SIFT descriptors
- element-wise encoding (hard assignment) on $k \sim 16$ visual words
- encoding now yields a residual vector rather than a scalar vote
- global sum pooling, ℓ^2 normalization

VLAD pipeline



- 3-channel RGB input \rightarrow 1-channel gray-scale
- set of ~ 1000 features \times 128-dim SIFT descriptors
- element-wise encoding (hard assignment) on $k \sim 16$ visual words
- encoding now yields a residual vector rather than a scalar vote
- global sum pooling, ℓ^2 normalization

VLAD pipeline



- 3-channel RGB input \rightarrow 1-channel gray-scale
- set of ~ 1000 features \times 128-dim SIFT descriptors
- element-wise encoding (hard assignment) on $k \sim 16$ visual words
- encoding now yields a residual vector rather than a scalar vote
- global sum pooling, ℓ^2 normalization

probabilistic interpretation

- if $p(X|C)$ is the likelihood of i.i.d observations X under a uniform isotropic Gaussian mixture model with component means C

$$p(X|C) \propto \prod_{x \in X} e^{-\frac{1}{2}\|x - q(x)\|^2}$$

- then the VLAD vector is proportional the gradient of $\ln p(X|C)$ with respect to the model parameters C

$$\mathcal{V}(X) \propto \nabla_C \ln p(X|C) = [\nabla_{c_1} \ln p(X|C), \dots, \nabla_{c_k} \ln p(X|C)]$$

- if we were to optimize C to fit the data X , then $\hat{\mathcal{V}}(X)$ would be the direction in which to modify C

probabilistic interpretation

- if $p(X|C)$ is the likelihood of i.i.d observations X under a uniform isotropic Gaussian mixture model with component means C

$$p(X|C) \propto \prod_{x \in X} e^{-\frac{1}{2}\|x - q(x)\|^2}$$

- then the VLAD vector is proportional the gradient of $\ln p(X|C)$ with respect to the model parameters C

$$\mathcal{V}(X) \propto \nabla_C \ln p(X|C) = [\nabla_{c_1} \ln p(X|C), \dots, \nabla_{c_k} \ln p(X|C)]$$

- if we were to optimize C to fit the data X , then $\hat{\mathcal{V}}(X)$ would be the direction in which to modify C

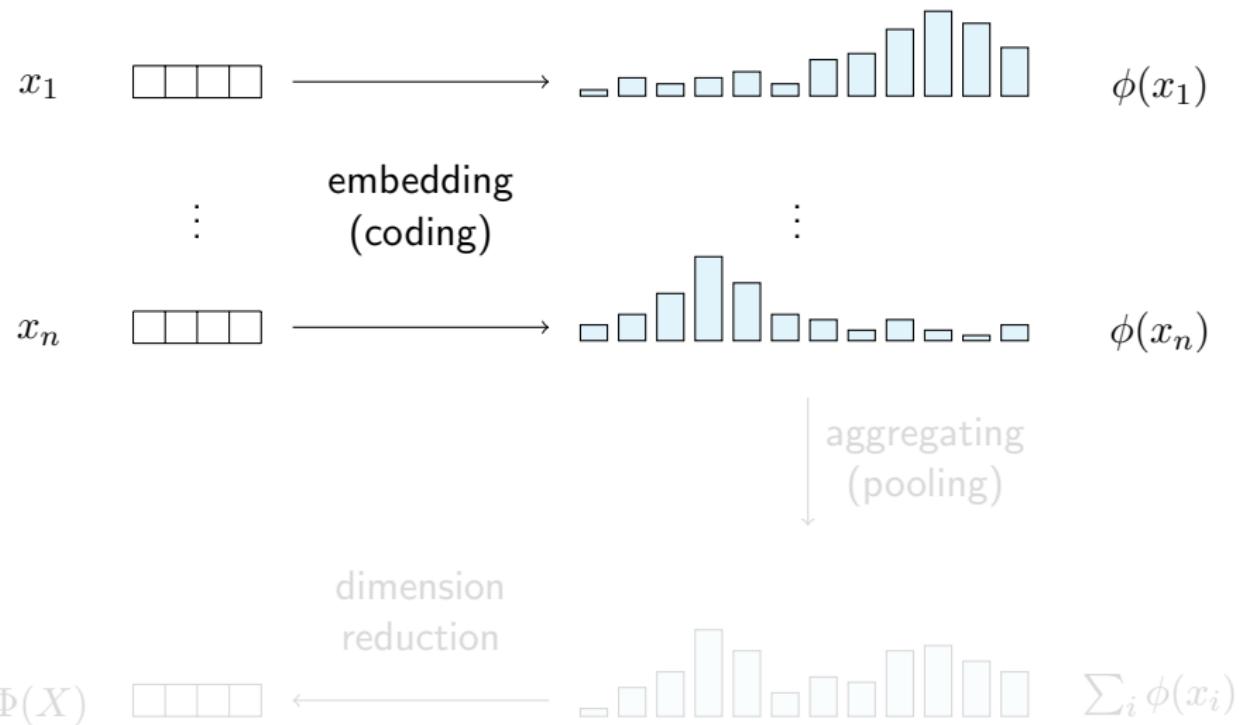
Fisher kernel

- the Fisher kernel generalizes to a non-uniform diagonal Gaussian mixture model

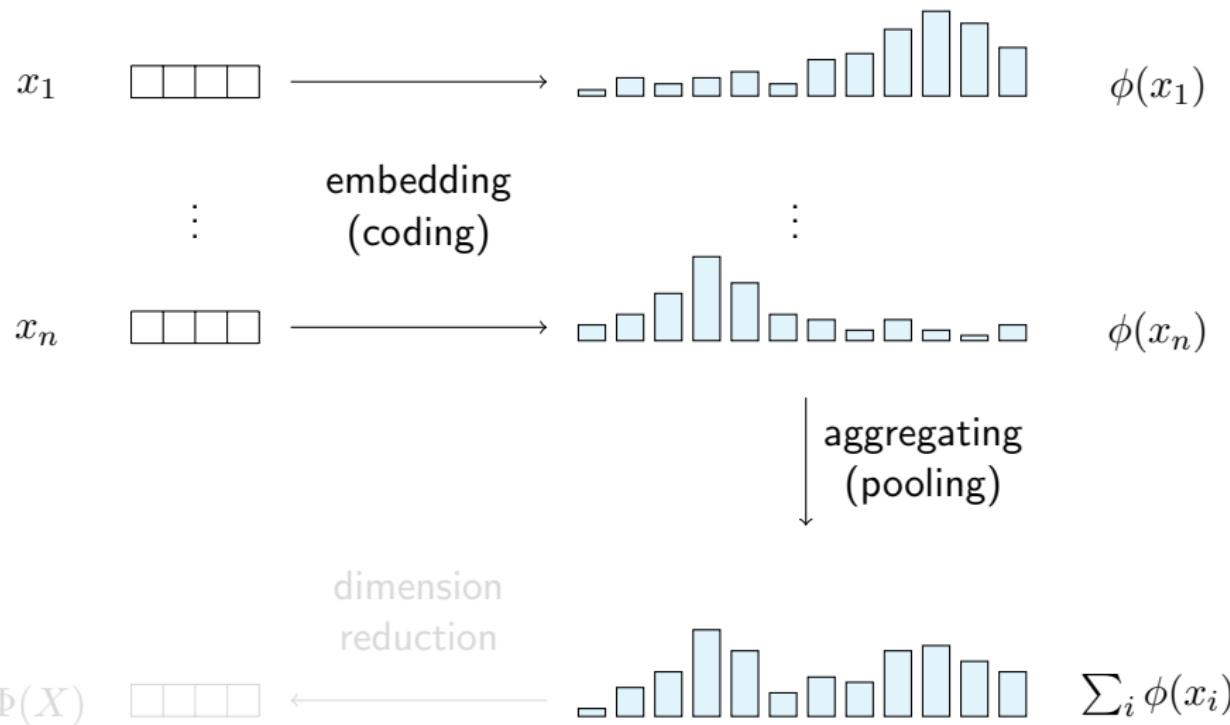
order statistics	parameter	model
0	mixing coefficient π	BoW
1	means μ	VLAD
2	standard deviations σ	Fisher

embeddings in general

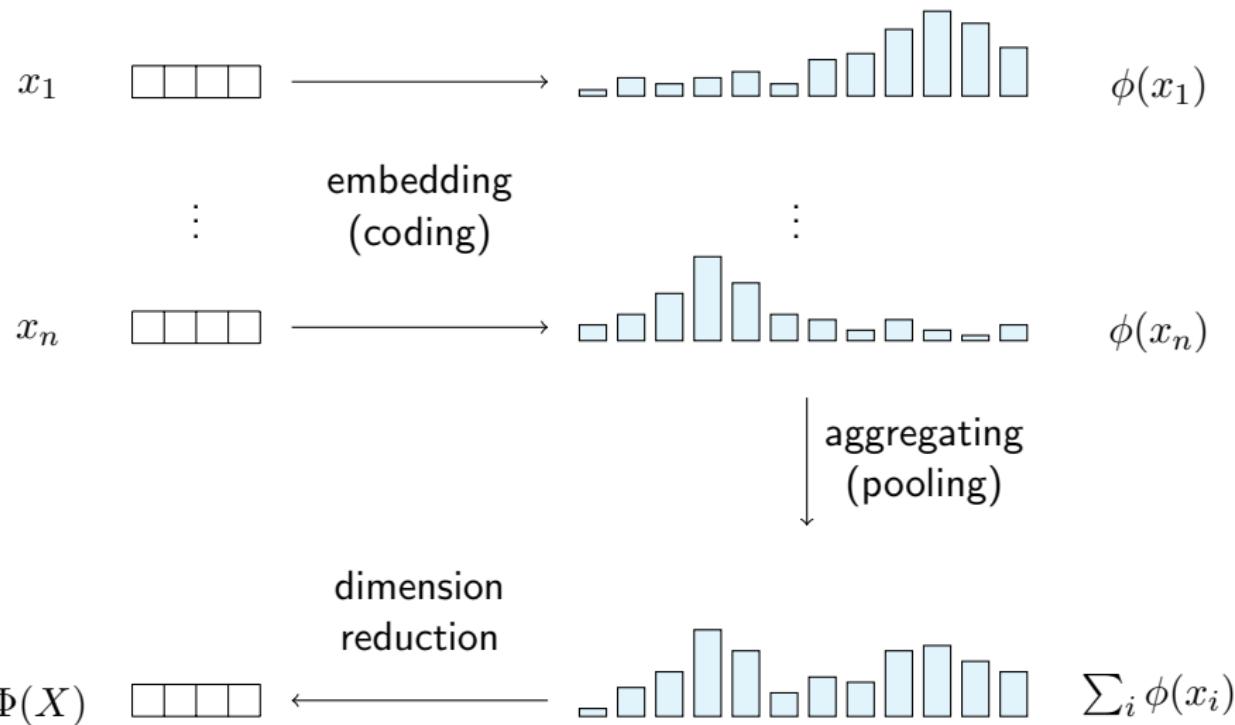
embeddings in general



embeddings in general

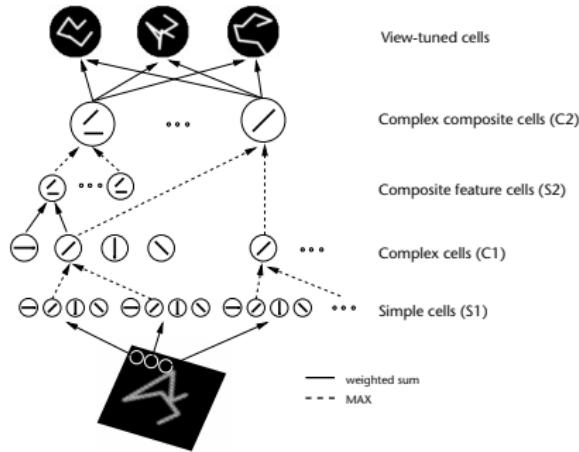


embeddings in general

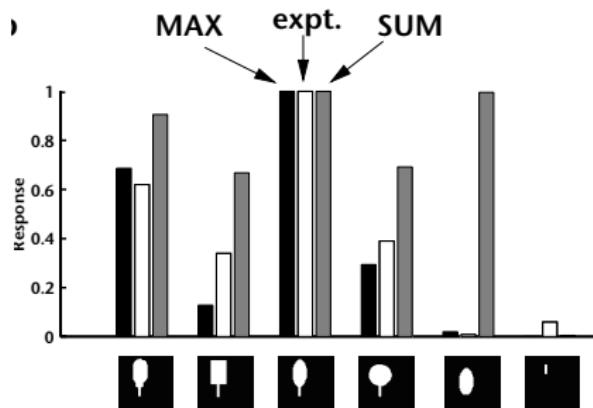


HMAX

[Riesenhuber and Poggio 1999]



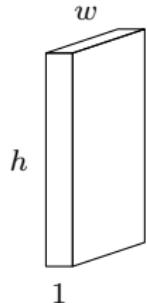
hierarchical model



sum vs. max pooling

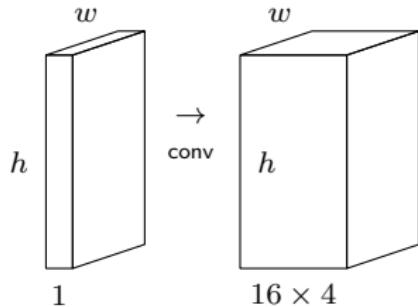
- computational model consistent with psychophysical data
- advocates non-linear max pooling

HMAX pipeline



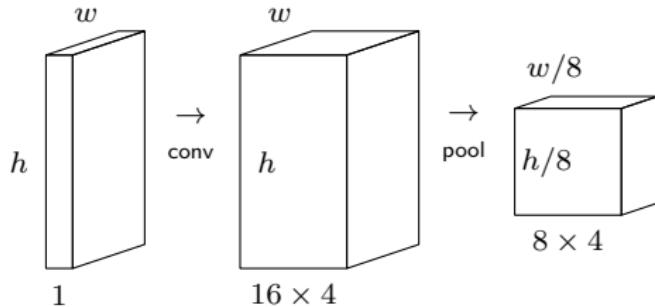
- 3-channel RGB input → 1-channel gray-scale
- apply filters at $16 \text{ scales} \times 4 \text{ orientations}$
- max-pooling over 8×8 spatial cells and over 2 scales
- convolutional RBF matching of input patches X to $k = 4072$ prototypes P (4×4 patches at 4 orientations) extracted **at random** during learning: activation $Y = \exp(-\gamma \|X - P\|^2)$
- global max pooling over positions and scales

HMAX pipeline



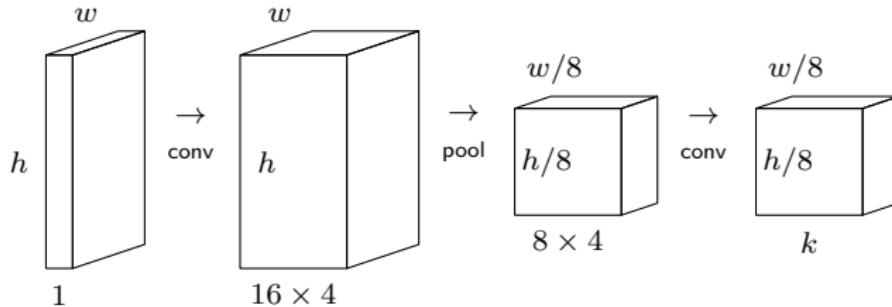
- 3-channel RGB input \rightarrow 1-channel gray-scale
- apply filters at 16 scales \times 4 orientations
 - max-pooling over 8×8 spatial cells and over 2 scales
 - convolutional RBF matching of input patches X to $k = 4072$ prototypes P (4×4 patches at 4 orientations) extracted **at random** during learning: activation $Y = \exp(-\gamma \|X - P\|^2)$
 - global max pooling over positions and scales

HMAX pipeline



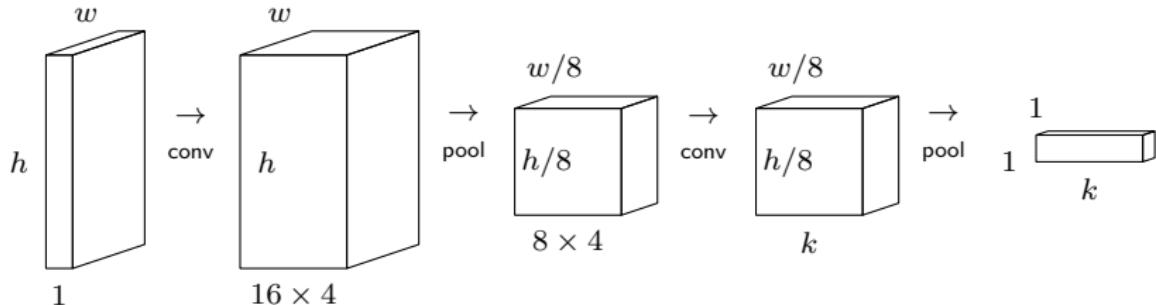
- 3-channel RGB input \rightarrow 1-channel gray-scale
- apply filters at 16 scales \times 4 orientations
- max-pooling over 8×8 spatial cells and over 2 scales
- convolutional RBF matching of input patches X to $k = 4072$ prototypes P (4×4 patches at 4 orientations) extracted **at random** during learning: activation $Y = \exp(-\gamma \|X - P\|^2)$
- global max pooling over positions and scales

HMAX pipeline



- 3-channel RGB input \rightarrow 1-channel gray-scale
- apply filters at 16 scales \times 4 orientations
- max-pooling over 8×8 spatial cells and over 2 scales
- convolutional RBF matching of input patches X to $k = 4072$ prototypes P (4×4 patches at 4 orientations) extracted **at random** during learning: activation $Y = \exp(-\gamma \|X - P\|^2)$
- global max pooling over positions and scales

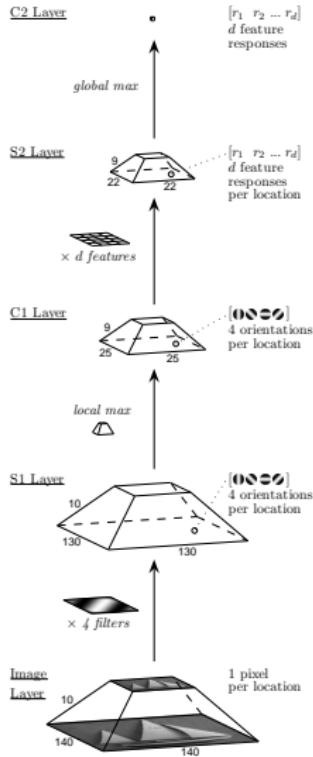
HMAX pipeline



- 3-channel RGB input \rightarrow 1-channel gray-scale
- apply filters at 16 scales \times 4 orientations
- max-pooling over 8×8 spatial cells and over 2 scales
- convolutional RBF matching of input patches X to $k = 4072$ prototypes P (4×4 patches at 4 orientations) extracted **at random** during learning: activation $Y = \exp(-\gamma \|X - P\|^2)$
- global max pooling over positions and scales

improvements

[Mutch and Lowe 2006]



- image pyramid
- S1 inhibition: non-maxima suppression over orientations
- strided C1 max pooling (50% overlap)
- C1 sparsification: dominant orientations kept

summary

- neuroscience background, convolution, Gabor filters
- texture analysis, frequency sampling, visual descriptors
- dense *vs.* sparse features
- gist, SIFT, HOG
- pooling Gabor filter responses *vs.* orientation histograms
- feature hierarchy, codebooks, encoding, pooling
- textons, BoW, VLAD, Fisher kernel, HMAX
- hard *vs.* soft encoding, max *vs.* sum pooling