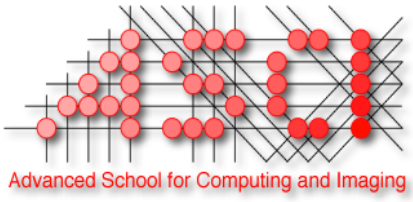
Computer Vision by Learning

Cees Snoek, UvA
Arnold W.M. Smeulders, UvA
Efstratios Gavves, UvA
Laurens van der Maaten Facebook



University of Amsterdam

Abstract

Computer vision has been revolutionized since the year 2000. Learning from examples is now leading. None of the methods for learning in computer vision is older than 15 years. In the course we will discuss methods of computing, invariance, and learning to distinguish objects.

The course is supplemented with practical work and is completed with an assignment.

Where and When

Wednesday 22nd feb to Tuesday 29th of feb

Lectures	09:30-12:15	C3.163
Lunch	12:15-13:30	on your own
Lab	13:30-17:00	B1.24.B

Lab

Lab Wednesday	Introduction vision by learning
Lab Thursday	Visualisation of filters and codes
Lab Friday	Style transfer learning
Lab Monday	Invariance and data augmentation
Lab Tuesday	Competition: bake your own

Each team of 2 persons hands in the Python notebooks per assignment completed with code and answers.

Deadline: **March 24th, 2017**

Email to: p.s.m.mettes@uva.nl

Overview Day 1

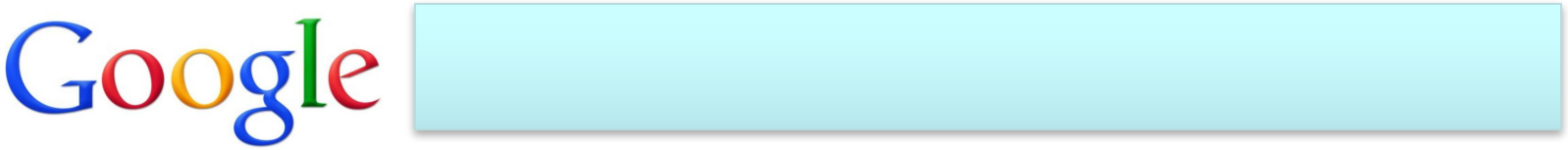
1. Introduction, types of concepts, tasks, knowledge as invariance
2. Observables, color, space, time, texture, Gaussian family
3. Invariance, the need for, color invariants, SIFT
4. BoW overview encode and classify
5. Encoders, sparse coding, autoencoders, Fisher vectors
6. SVM, separability, kernels
7. Finale: BoW vs CNN

1. Introduction

How is a computer capable of converting digital camera recordings to a notion what is on the picture? How does it know how a boat looks like a boat, when there are so many different types and so many different views? And what is harder to recognize, a fork or a knife? A glass or a plate? We discuss concepts and aspects and how a computer can learn to recognize these things in an image.

What does Google know?

When you type in a question in Google, it goes through its records to see what other people have used for an answer.



Brilliant move.

They (used to reproduce) what other people know.

We start from what *we* know

How do we convert light intensity patterns into meaning?

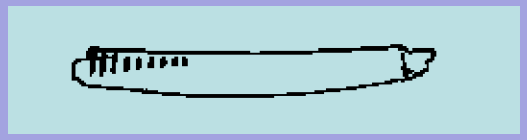
How does your eye, any eye, learns to recognize this?

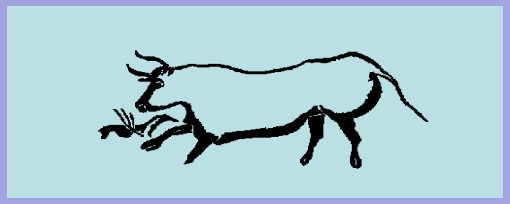
Mankind’s great invention

Once upon a time, someone pointed at

and said “there”...

The invention 1. of visual recognition and 2. of language:

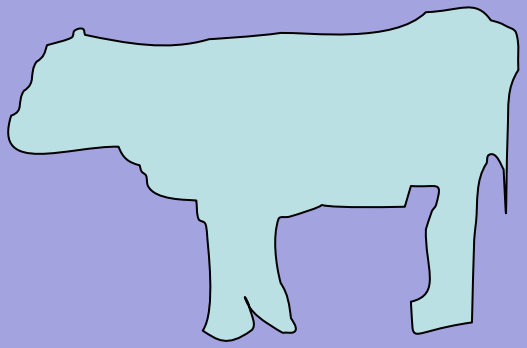




Vision and words

Later, “there” became “cow”. Things got their names.

How is this done? Most people assume outline & recognize.



Learn to recognize objects

What knowledge is needed for proper segmentation?

When is segmentation really needed?

Quiz: What actions require a. segment, b. recognize, c. both?

1. catch?

2. avoid?

3. cuddle?

4. grab?

5. puzzle?

Learning in Computer Vision

What is *tea*?

This is the semantic gap. Learn, not model.

Smeulders PAMI 2000

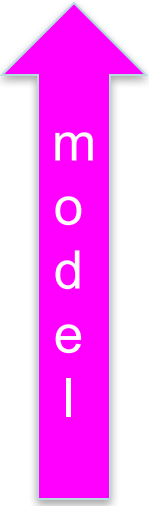
Things learned in Computer Vision

A concept is a named entity *bicycle*, *2 persons* or
a mass-good *grass*, *ants*, *tea*, *measles* or
a labeled scene *mountainous*, *birthday*

An aspect is a state of *happy*, *dried*, *atherosclerosis*

Quiz 1: Name *tea* - subtypes.
2: Name visual tasks per subtype?

Tasks in Computer Vision



Learn concepts given labeled examples.

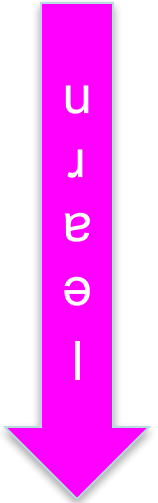
Localize concepts given labeled examples.

Count concepts

Search for examples similar to this.

Explain evidence for concepts.

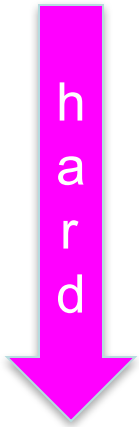
Estimate variance intrinsic or extrinsic to concept.



Generalize aspects over concepts.

Numbers of examples

Number of examples per concept: capture the variance.



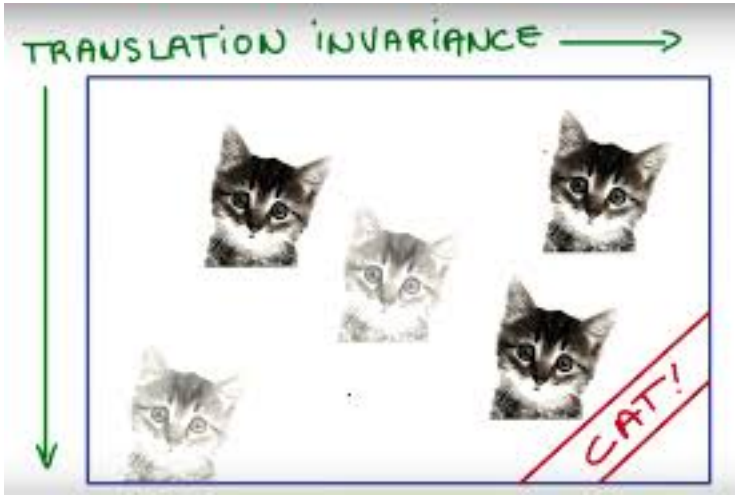
50	gives a good insight in common variations
1000	covers even rare variations
1	search for look-alikes
0	consult sources of knowledge
7	what is structural, what is accidental variation?

Let's understand this order.

Knowledge is …

Human knowledge is laid down in invariances. Statements which are always true, regardless of accidental conditions.

Learning starts from an analysis of variances.



2. Observables

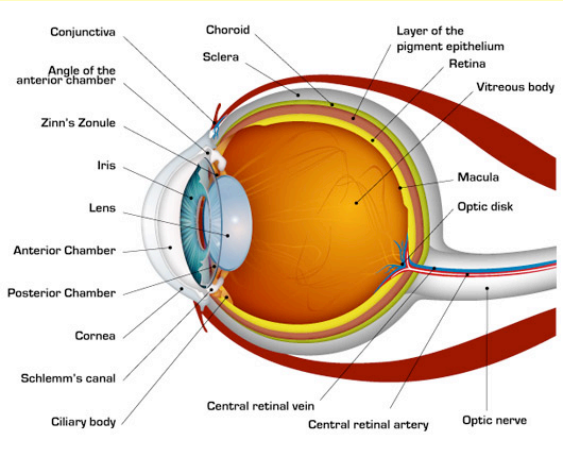
We recapitulate color, spatial, temporal and texture observables.

Color is powerful as pixel observations lead to object intrinsic information separated from image accidental information.

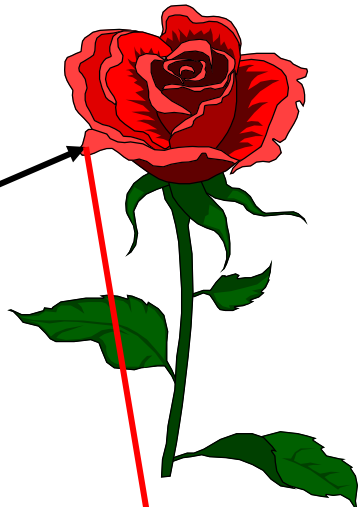
Spatial observables form the Taylor expansion of the image. We discuss the Gaussian implementation including the temporal.

Finally, texture is defined by Gabor filters.

The observables are brought together in the Gaussian family.



Formation of image values

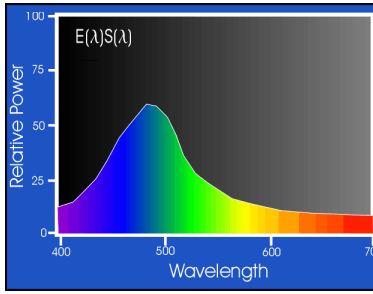


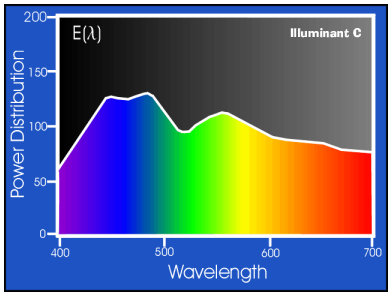
$S(\lambda)$

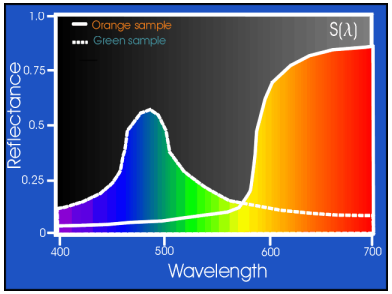
$r(\lambda)S(\lambda)$

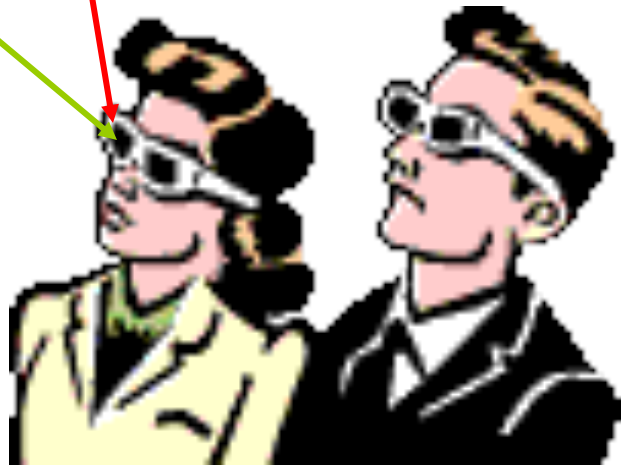
object reflection

Source



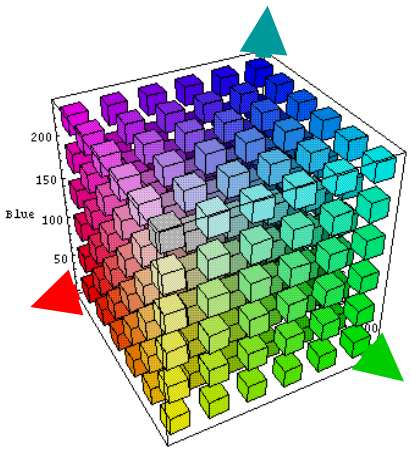


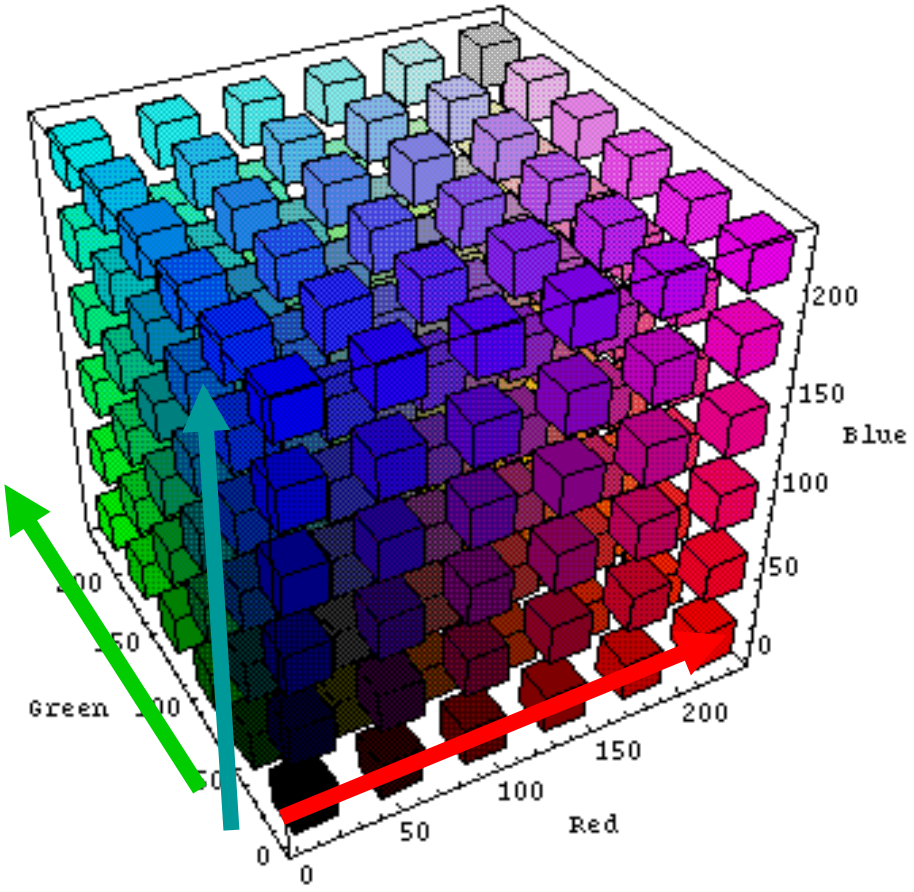




E = (R,G,B)

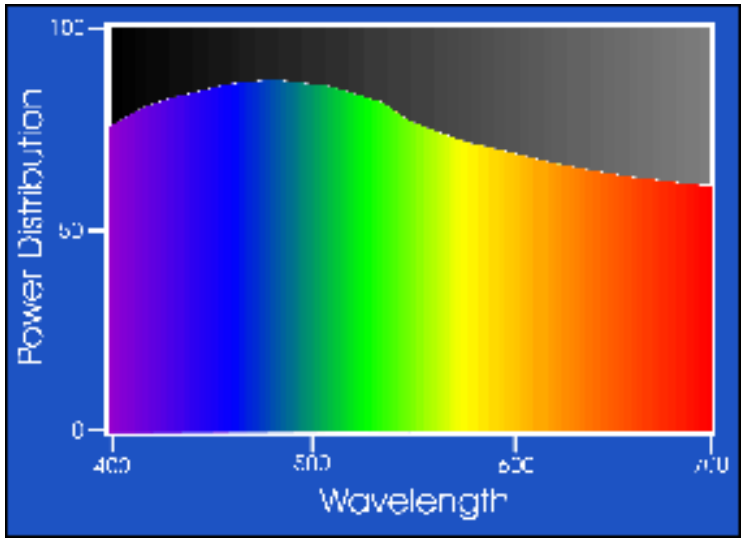
$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} \int_\lambda e(\lambda)\rho(\lambda)f_R(\lambda)d\lambda \\ \int_\lambda e(\lambda)\rho(\lambda)f_G(\lambda)d\lambda \\ \int_\lambda e(\lambda)\rho(\lambda)f_B(\lambda)d\lambda \end{pmatrix}$$





HSI = Human sensation of light

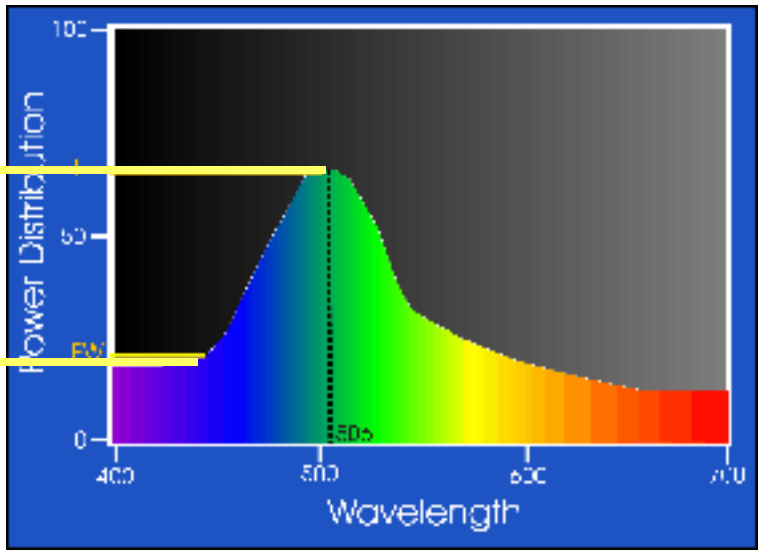
Hue:	dominant wavelength	$\lambda(E_H)$
Saturation:	purity of the colour	$(E_H - E_W)/E_H$
Intensity:	brightness of the colour	E_W



white light

E_H

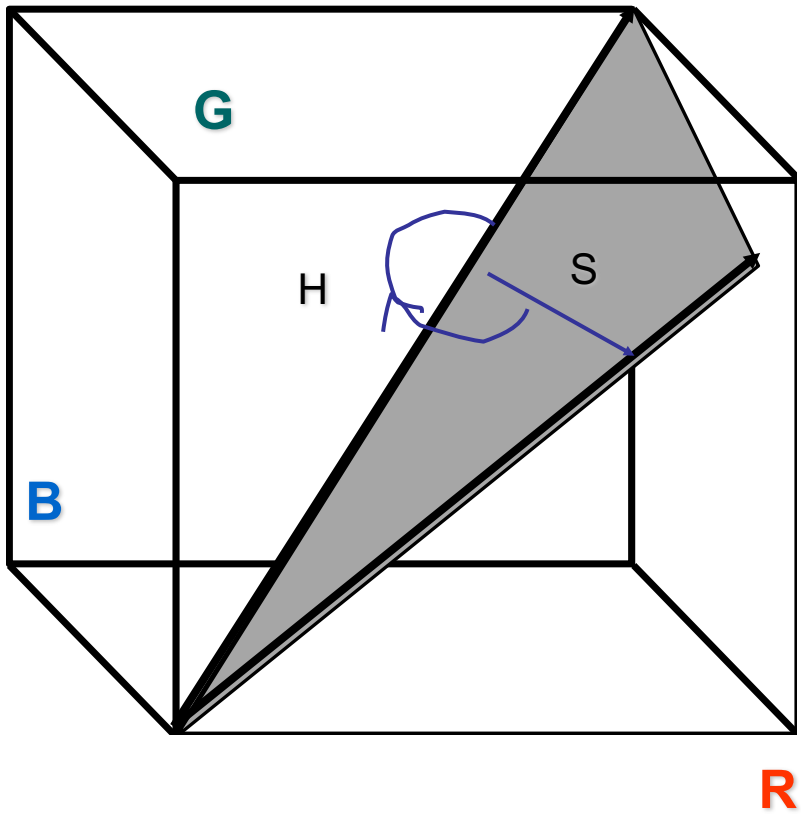
E_W



green light

HSI in E - space

$$\begin{pmatrix} H \\ S \\ I \end{pmatrix} = \begin{pmatrix} arctg\left(\dfrac{\sqrt{3}(G-B)}{(R-G)+(R-B)}\right) \\ 1-\dfrac{\min(R,G,B)}{R+G+B} \\ R+G+B \end{pmatrix}$$

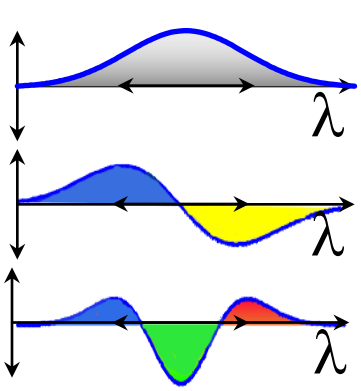


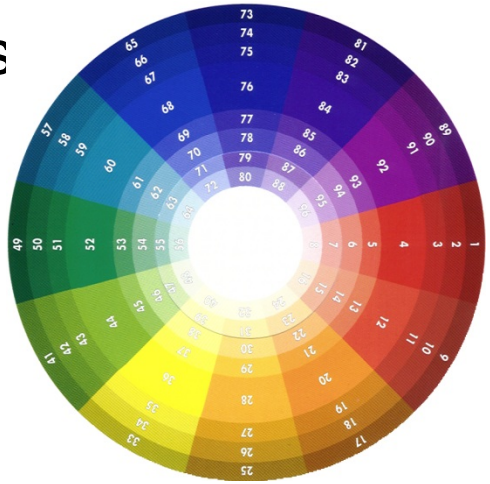
H & S are illumination invariant body properties, separating the accidental recording conditions from the intrinsic body condition.

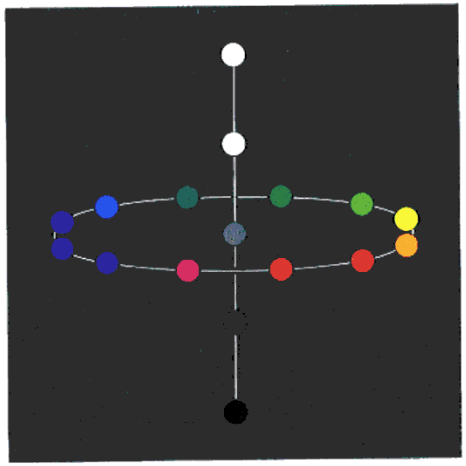
Opponent colors

Human perception combines (R,G,B) respons
of the eye in opponent colors

$$\begin{pmatrix} \text{Luminance} \\ \text{BlueYellow} \\ \text{PuperGreen} \end{pmatrix} = \begin{pmatrix} R+G+B \\ \frac{1}{2}(R-G) \\ \frac{1}{4}(2B-R-G) \end{pmatrix}$$



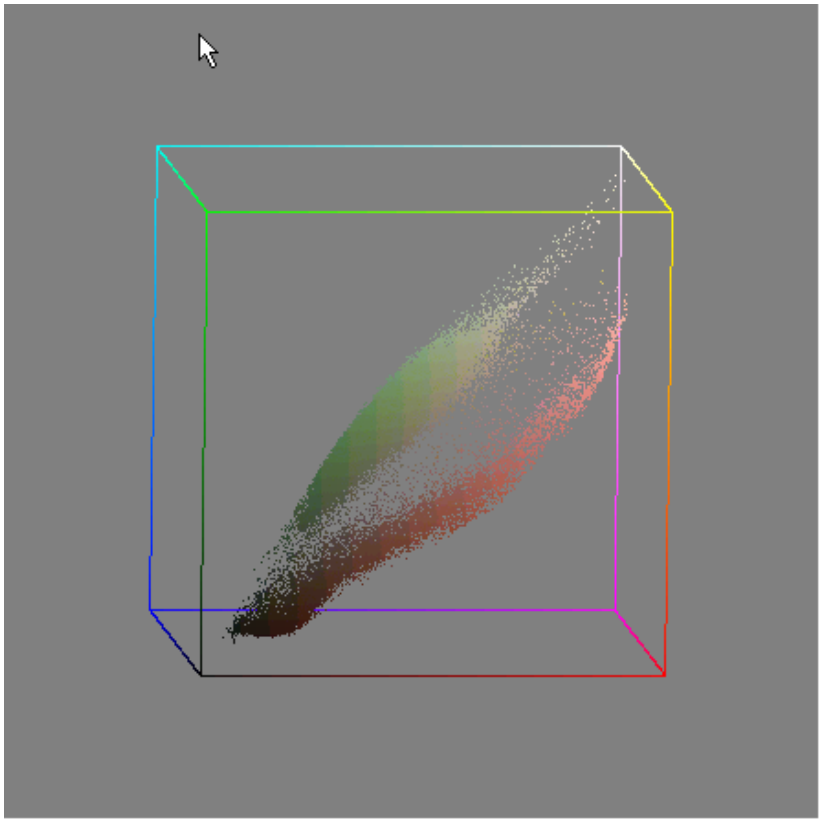




as it maximizes the contrast.

Pixel-level

Pixel values have accidental variation due to body reflection.



Cancel them out, first.

Analysis

So far we have talked about how light makes images.
Now we want to turn it around: we learn off images.

Which properties do we measure?
How do we measure properly?

The Taylor expansion for light is:

$$\hat{E}(x) \approx \sum_{j=0}^{n} \frac{1}{j!} (x \nabla_x)^j \hat{E}(x)$$

Sampling the light

The Taylor expansion for light is:

$$\hat{E}(x) \approx \sum_{j=0}^{n} \frac{1}{j!} (x \nabla_x)^j \hat{E}(x)$$

For discretely sampled signal use Gaussian filters

$$\nabla_x{}^j \hat{E} = \frac{\partial^j \hat{E}}{\partial^j x} = E * G_{x^j}^{x_0, \sigma_x}$$

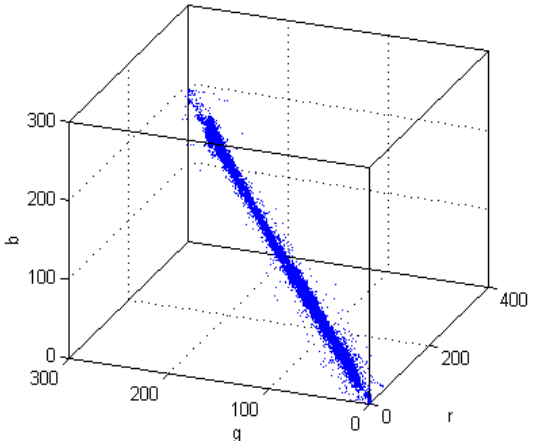
Among the class of linear filters, Gauss is separable by dimension, rotationally uniform, sloping to zero, adds no maxima and so on. It is the preferred brand of local filters.

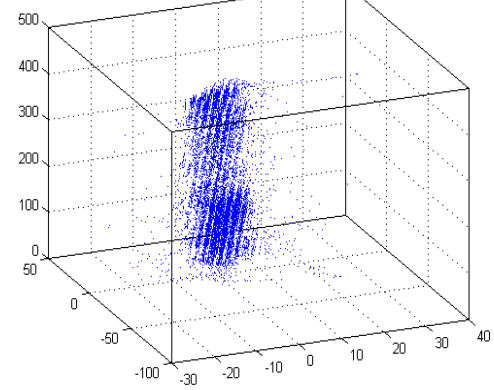
Color Gaussians

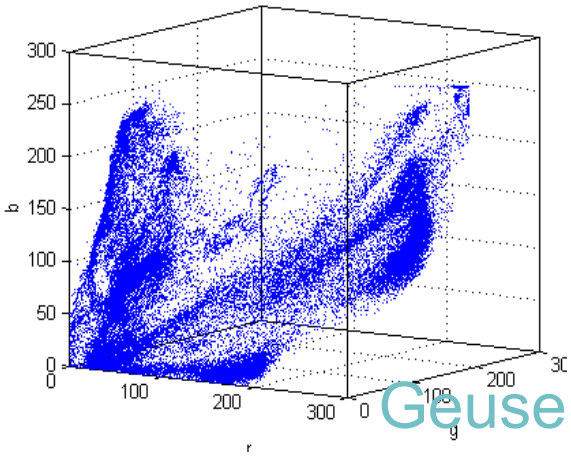
$$\begin{pmatrix} E \\ E_\lambda \\ E_{\lambda\lambda} \end{pmatrix} = \begin{pmatrix} 0.06 & 0.63 & 0.27 \\ 0.30 & 0.04 & -0.35 \\ 0.34 & -0.60 & 0.17 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

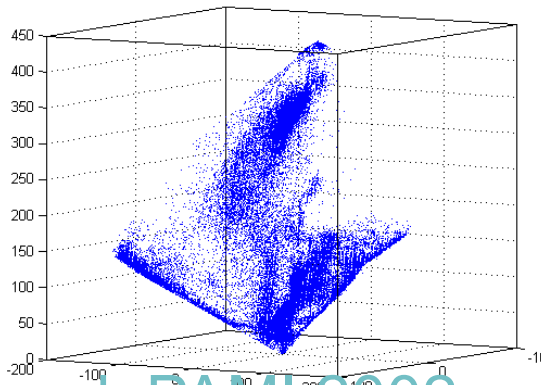
(R,G,B)-pdf

(E_0,E_λ,$E_{\lambda\lambda}$)-pdf





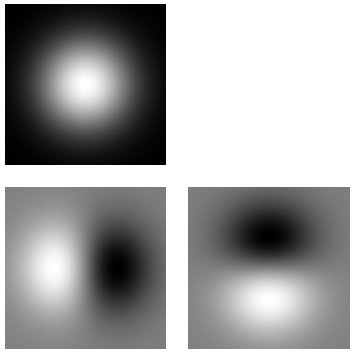




Geusebroek PAMI 2002

Spatial Gaussians

For spatial filters, this is Gauss in zero and first order:

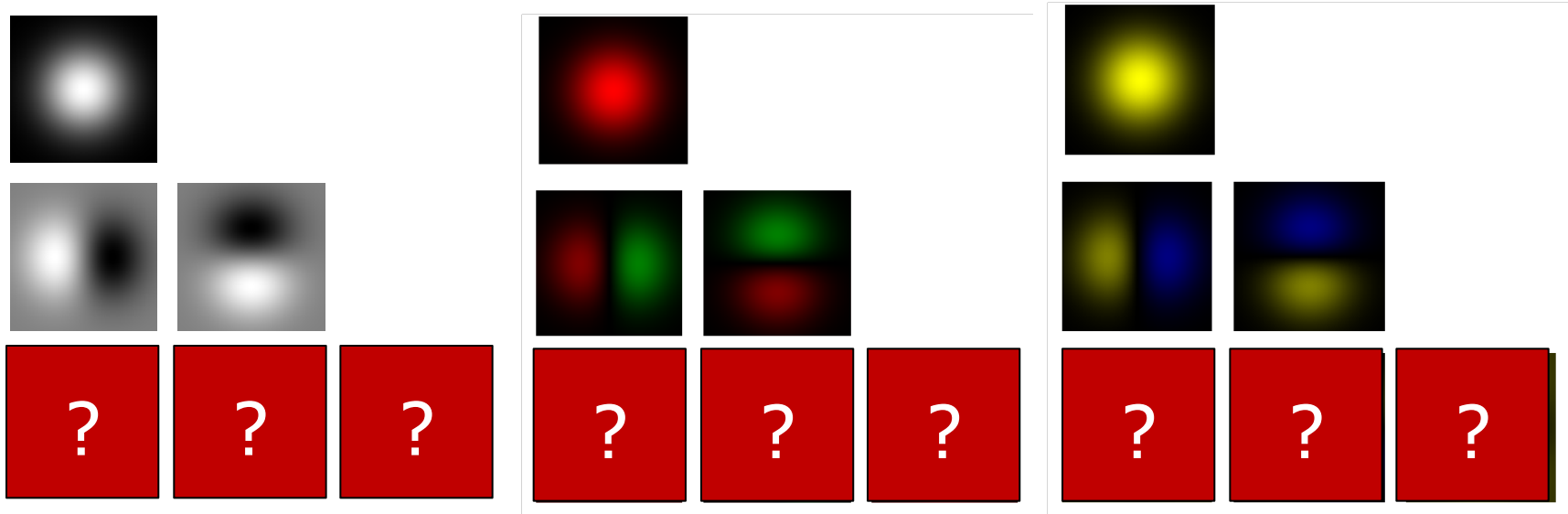


Fast approximate recursive implementation:

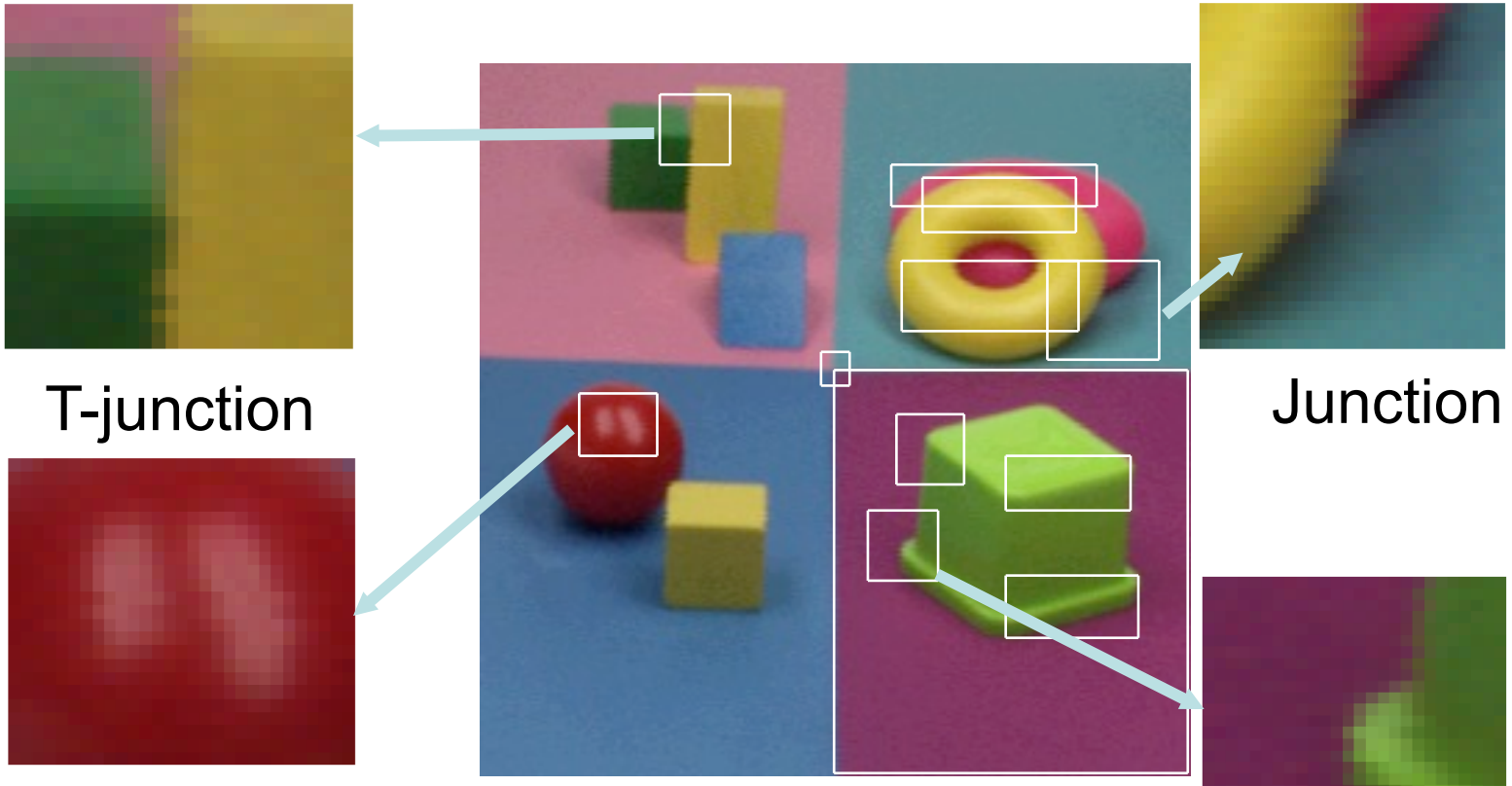
Geusebroek, Van de Weijer & Smeulders 2002

Quiz: Draw 2nd color/spatial order

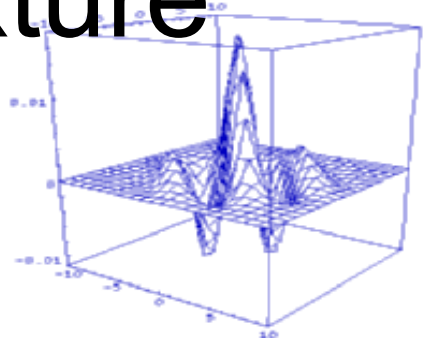
Gaussian in zero and first order and their receptive fields:



Differentials are everywhere



Gaussian - Gabor texture



The 2D Gabor function with parameters: *u, v, σ*:

$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\delta^2}} e^{2\pi j(ux+vy)}$$

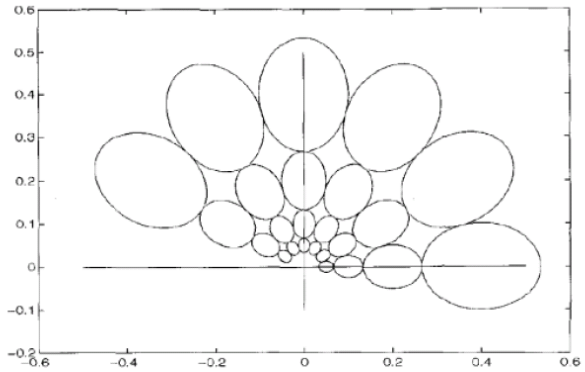


Fig. 1. The contours indicate the half-peak magnitude of the filter responses in the Gabor filter dictionary. The filter parameters used are $U_h = 0.4$, $U_l = 0.05$, $K = 6$, and $S = 4$.

Gabor for texture in Fourier-space

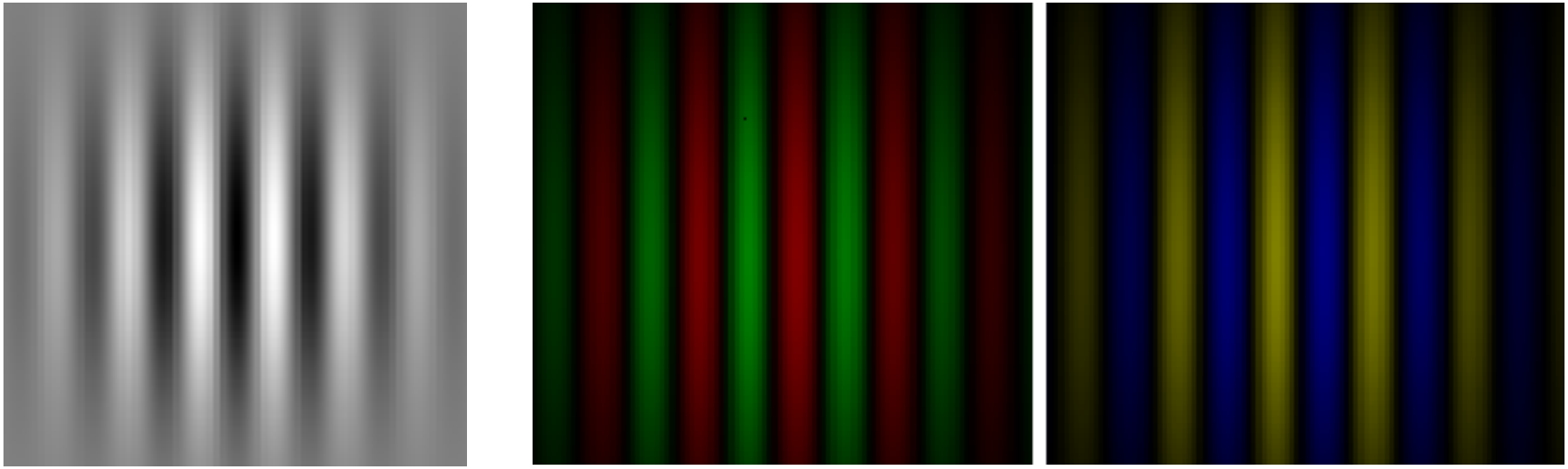
$$a = (U_h/U_l)^{-\frac{1}{S-1}}, \quad \sigma_u = \frac{(a-1)U_h}{(a+1)\sqrt{2\ln 2}},$$

$$\sigma_v = \tan\left(\frac{\pi}{2k}\right)\left[U_h - 2\ln\left(\frac{\sigma_u^2}{U_h}\right)\right]\left[2\ln 2 - \frac{(2\ln 2)^2\sigma_u^2}{U_h^2}\right]^{-\frac{1}{2}}, \quad (4)$$

Manjunath & Ma

Gaussian - Gabor texture

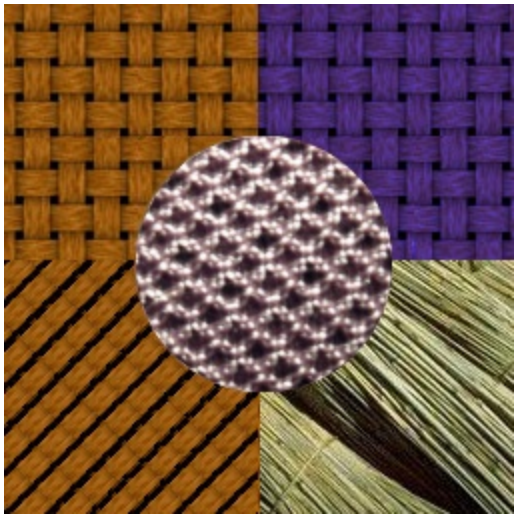
The receptive fields for (u, v) measured locally

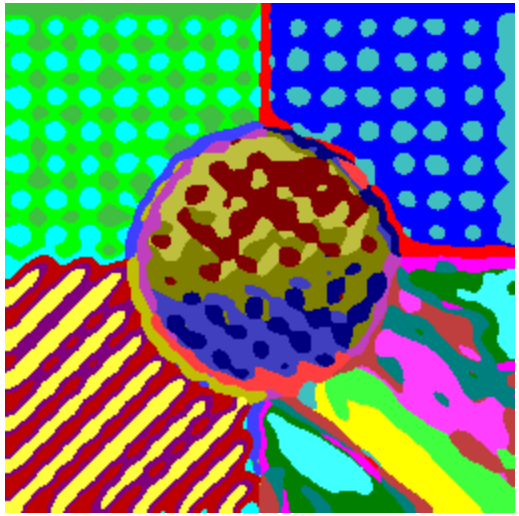


Grey and opponent color feature sets.

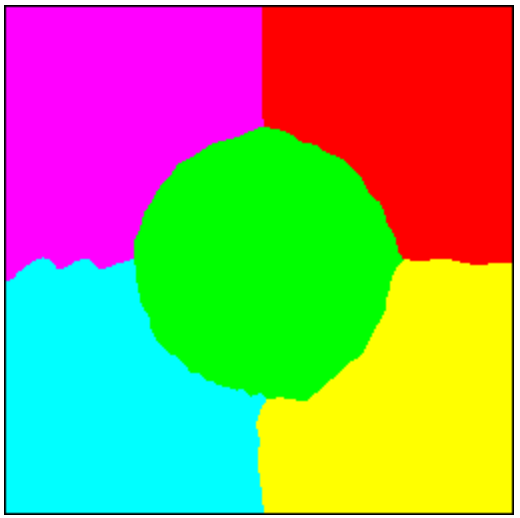
Hoang SP 2003

Gaussian - Gabor texture





K-means cluster of RGB

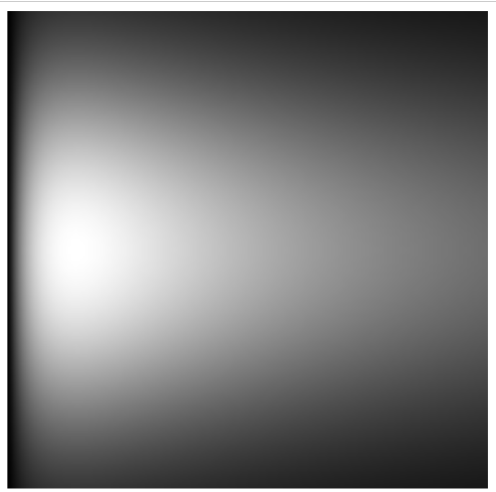


K-means cluster Gabor opponent

Hoang ECCV 2002

Gaussian temporals

Gaussian equivalent over x and t:



zero order

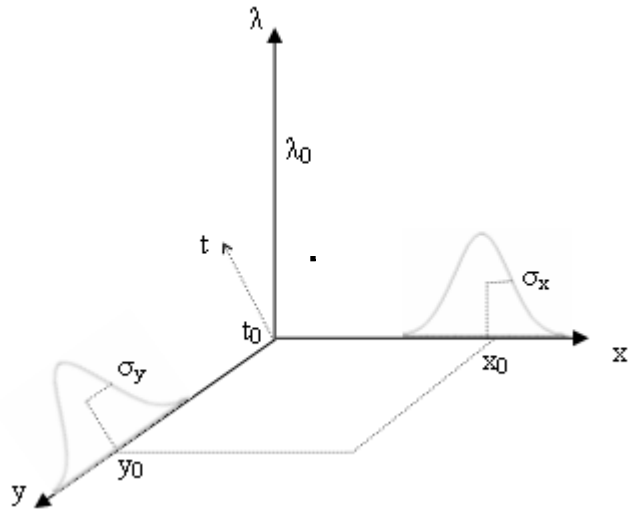
first order over t

Gaussian: the complete family

$$\hat{E}(x) \approx \sum_{j=0}^{n} \frac{1}{j!} (x \nabla_x)^j \hat{E}(x)$$

The basic visual observables are:

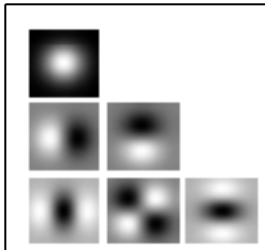
$$G(x, y, \lambda, t)$$

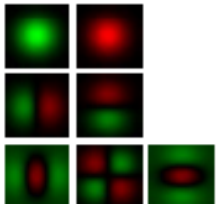


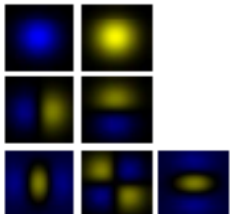
$$G^{x_0;\sigma_x} * G^{y_0;\sigma_y} * G^{\lambda_0;\sigma_\lambda} * G^{t_0;\sigma_t}$$

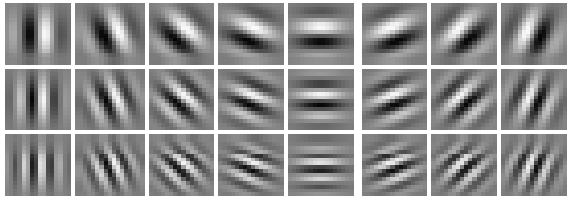
Gaussian: the complete family

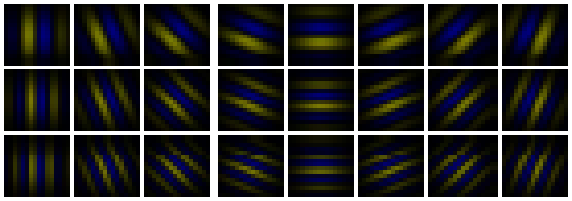
Among the class of linear filters, only Gauss is separable by dimension when rotationally uniform, and adds no maxima.

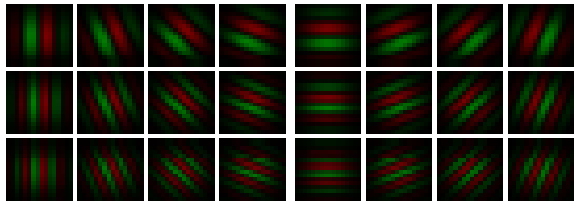


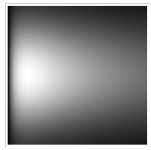


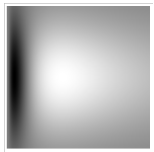












All observables up to first order color, second order spatial scales, eight frequency bands & first order in t.

Good observables

1. Good features ignore all *irrelevant* variations,
   accidental, recording specific, individual variations.
2. Good features capture all *distinguishing* variations,
   pairs-of-class-specific variations.
3. Good sampling provides all relevant group information
   either by random sampling or stratified sampling.
4. Good features have a good repeatability,
   and produce little noise themselves.

Good observables: note to 4.

How errors ruin your features:

For sums of two observables: $\Delta x = \Delta x_1 + \Delta x_2$

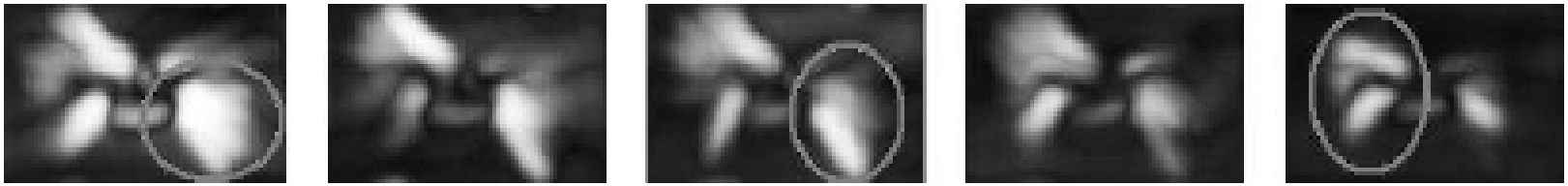
Products of two observables: $\frac{\Delta x}{x} = \frac{\Delta x_1}{x_1} + \frac{\Delta x_2}{x_2}$

a. Close to 0, beware of division!
b. Invariants are usually a division!
a. + b. : Beware of using invariants!

Good observables ≅ easy algorithm

Periodicity:

Detect periodic motion by one steered filter:



Deadly simple algorithm…

Burghouts TIP 2006

3. Invariants

Invariance aims to exclude all irrelevant variations.

Color invariants are powerful for everything related to illumination, and in the end simple.

The quality of invariant features: invariance + discrimination.

The need for invariance

There are a million appearances to one object

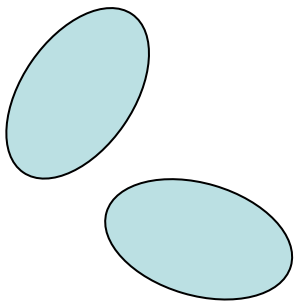
The same part of the same shoe does not have the same appearance in the image. This is the sensory gap.

Remove unwanted variance as early as you can.

The need for invariance

A feature g is invariant under condition (transform) W caused by accidental conditions at the time of recording, iff g observed on equal objects t_1 and t_2 is constant:

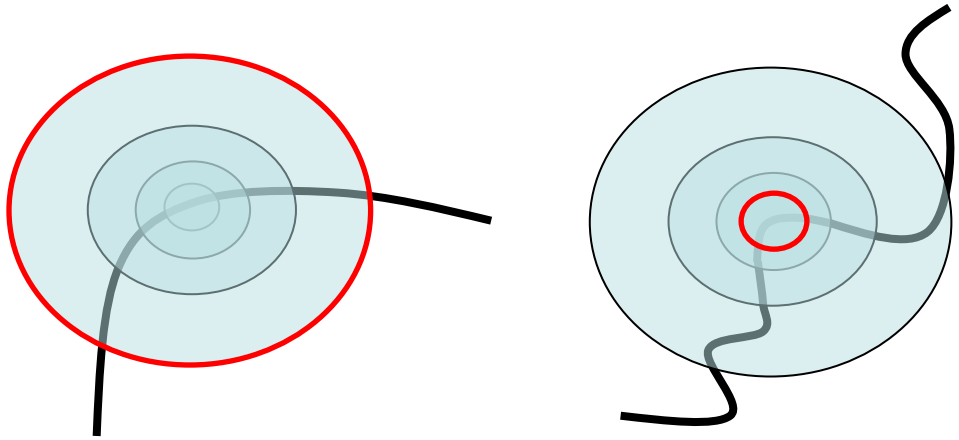
$$t_1 \overset{W}{\sim} t_2 \Rightarrow f_{t_1} = f_{t_2}$$



Length of long axis / short axis independent of scale and rotation.

Quiz: scale invariant detection

What properties are invariant to observation scale?



Scale invariants are at least dimensionless numbers:

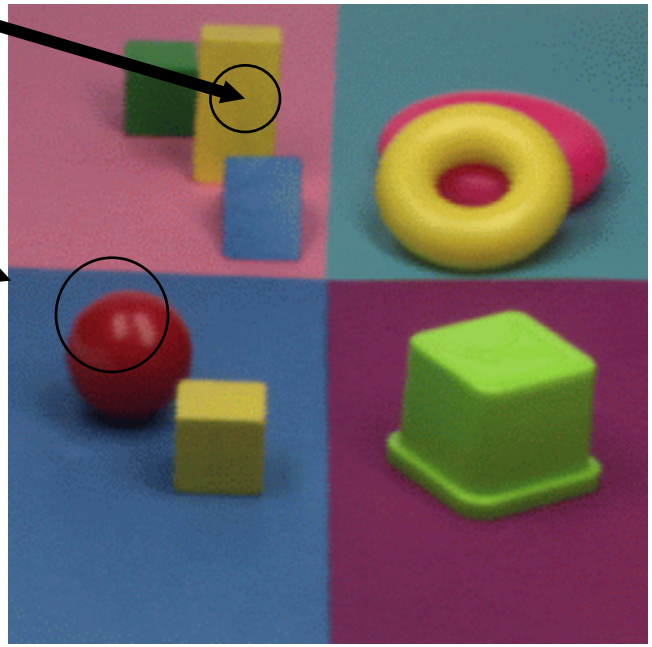
+ The ratio of the long and short axes.
+ Corners.
+ Length squared divided by area.

Color invariance: reflectance

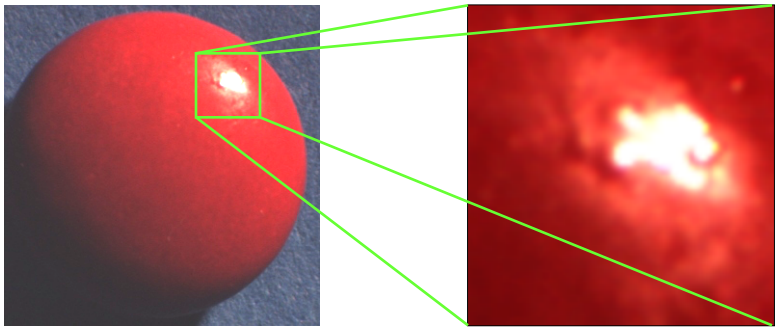
Reflectance model [Shafer]

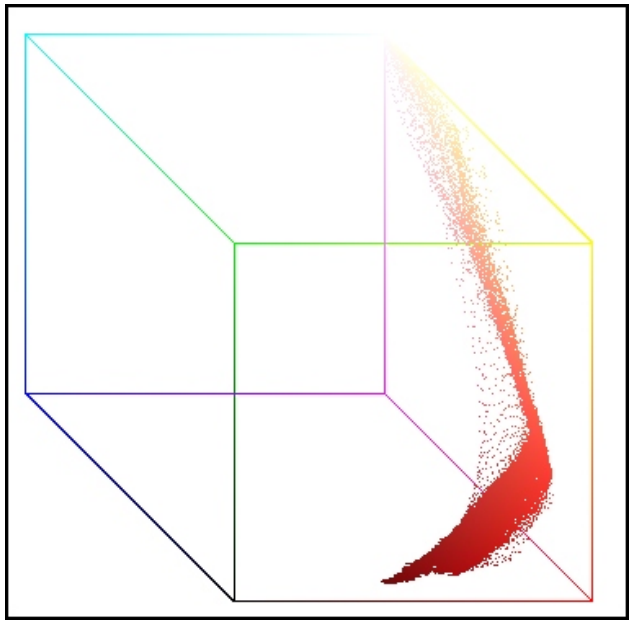
body = $m_b(\vec{n},\vec{s})\int_\lambda e(\lambda)c_b(\lambda)f_C(\lambda)d\lambda$

surface = $m_s(\vec{n},\vec{s},\vec{v})\int_\lambda e(\lambda)c_s(\lambda)f_C(\lambda)d\lambda$



E is viewpoint variant





The surface reflection term:

$$m_s(\vec{n},\vec{s},\vec{v})\int_{\lambda} e(\lambda)c_s(\lambda)f_C(\lambda)d\lambda$$

reduces to simpler form of Phong:

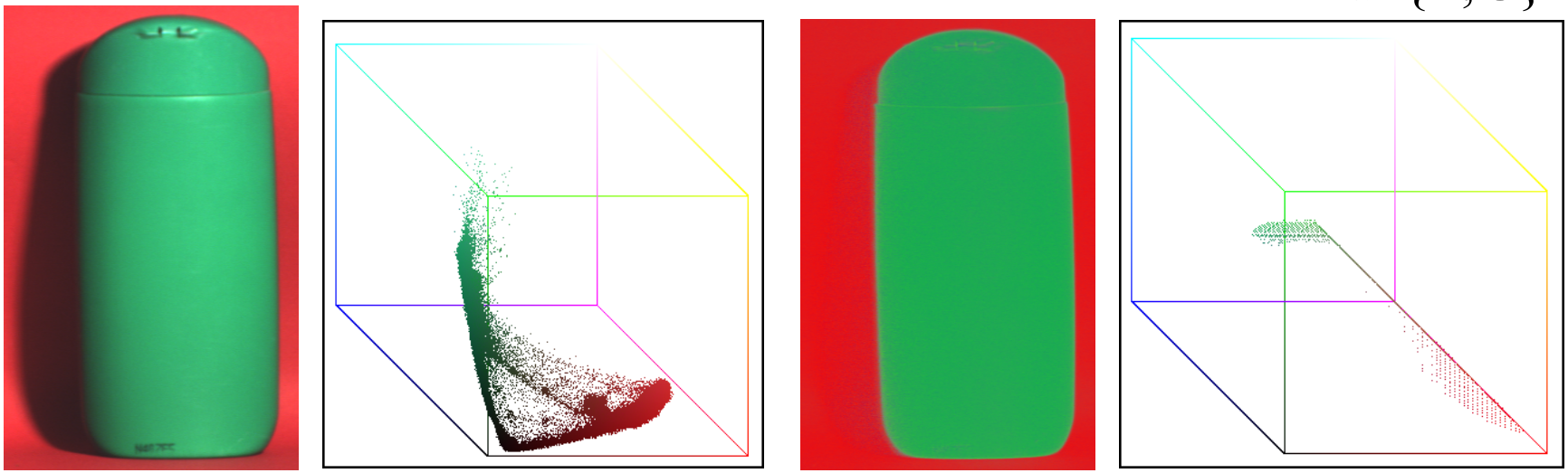
$$m_s(\vec{n},\vec{s},\vec{v}) \propto \cos^n(\alpha(\vec{n},\vec{s},\vec{v}))$$

C is viewpoint invariant

$$c_1(R,G,B) = \arctan\frac{R}{\max\{G,B\}}$$

$$c_2(R,G,B) = \arctan\frac{G}{\max\{R,B\}}$$

$$c_3(R,G,B) = \arctan\frac{B}{\max\{R,G\}}$$



E space

C space

Gevers TIP 2000

Differential invariants **C'**, **W'**, **M'**

C' is for matte objects and uneven white light:

$$C_\lambda = \frac{E_\lambda}{E} \qquad C_{\lambda\lambda} = \frac{E_{\lambda\lambda}}{E} \qquad C_{\lambda x} = \frac{E_{\lambda x}E - E_\lambda E_x}{E^2}$$

W' is for matte planar objects and even white light:

$$W_x = \frac{E_x}{E} \qquad W_{\lambda x} = \frac{E_{\lambda x}}{E}$$

M' is for matte objects and monochromatic light:

$$N_{\lambda x} = \frac{E_{\lambda x}E - E_\lambda E_x}{E^2}$$

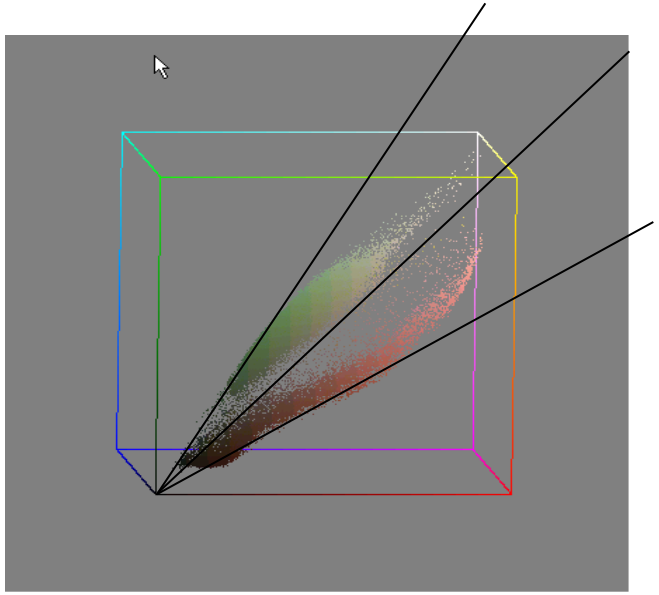
Geusebroek PAMI 2002

Invariance & Discrimination

Total loser

The most invariant feature is the value “42”.

Desired invariance ↔ undesired loss of discriminative power.



Retained discrimination

	shadows	shading	highlights	ill. intensity	ill. color
E	-	-	-	-	-
H	+	+	+	+	-
W & W'	-	+	-	+	-
C & C'	+	+	-	+	-
M & M'	+	+	-	+	+
L	+	+	+	+	-

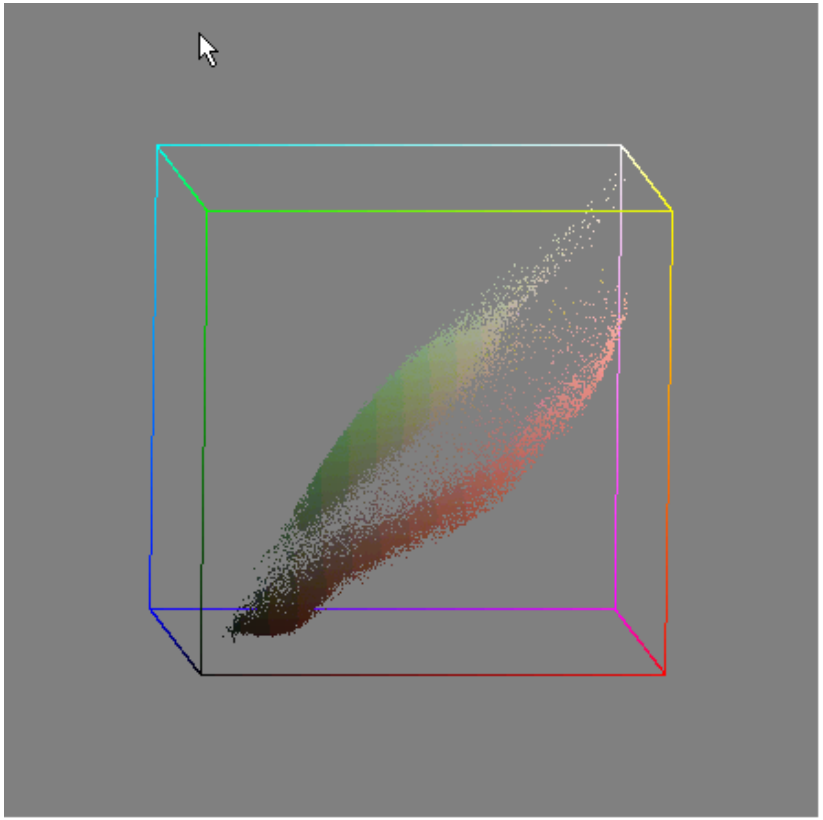
Retained from 1000 colors $\sigma = 3$:

E	990
H	315
W'	995
C'	850
M'	900

Geusebroek PAMI 2003

Quiz: a. What sources of variance b. Find ways to get invariance

This is an orbit.

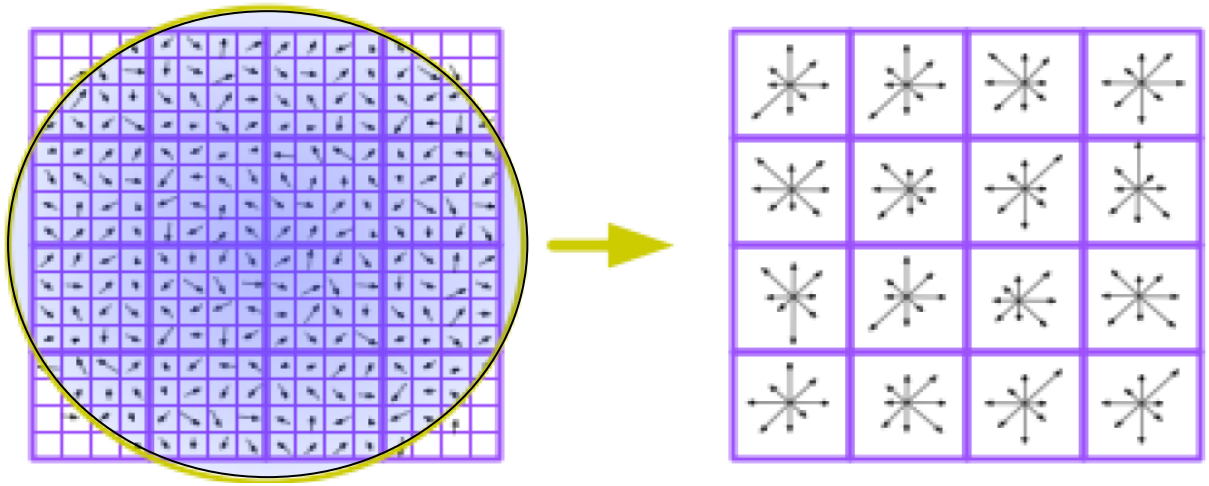


SIFT

For 4x4 patches, find 4x4 significant gradients > threshold.

Find orientation spectrum by #significant edges per direction.

8 orientations in 16 patches, normalize = 128D SIFT feature.

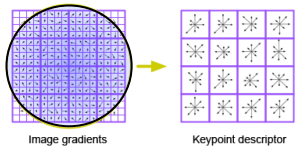


Lowe IJCV 2004

Image gradient directions

histogram = SIFT

SIFT invariances



1. Intensity invariant due to orientation of edge only.
2. Surface roughness invariant due to thresholded gradients.
3. Occlusion invariant [when visible] due to locality.
4. Viewpoint mildly invariant by using dominant orientation.
5. Rotation hardly invariant (*normalizing* dominant direction)
6. Scale hardly invariant (*maximize* response over scales)
7. Translation not invariant (*collect* response over region)

while still

Discriminative among objects due to high dimensionality

(*enhance* by invariant colors [Van der Sande PAMI 2010])

4. Bags of Words

SIFT solves illumination, roughness, occlusion, viewpoint to 30^{o}.

invariance left to cover	remedy
Viewpoint beyond 30^{o}	*record* many views
Rotation	*normalize* orientation, *collect* points
Scale	*maximize* over scale views
Translation	*collect* many points in a bag

Bag of words is a multi-stage code & classify process aimed at separating variability of feature values from invariant labels.

Capture the pattern in each patch

Measure the pattern in a patch with abundant features.

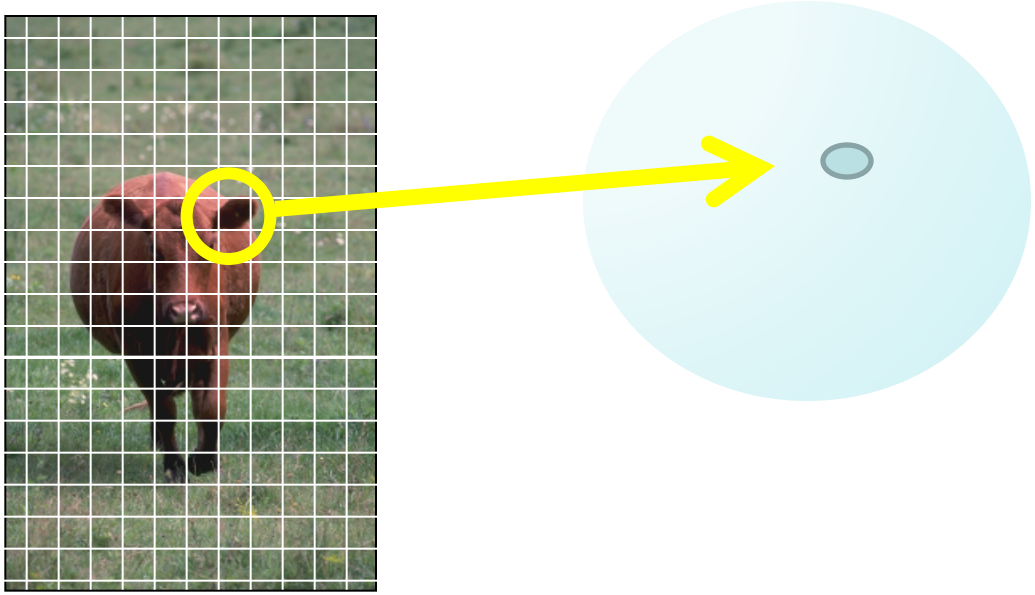
SIFT is good, cSIFT is better.

Gauss family will also do it.

Normalized is better.

More patches is better.

More features is better.

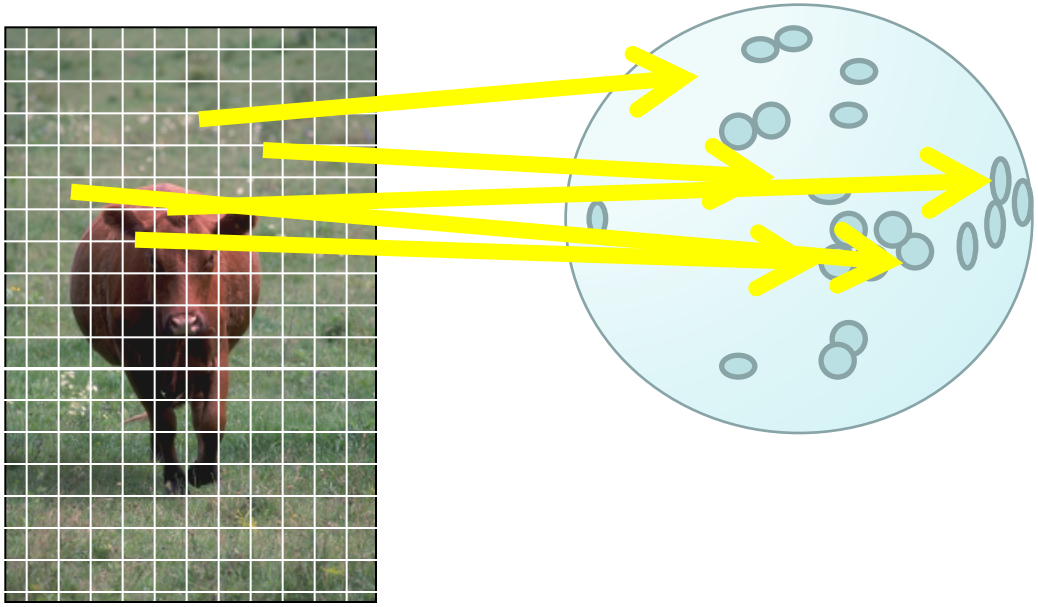


Sample many patches

Sample the patches in the image. Salient-sampling 2**10.

Densely 2**18.

Salience is good, dense better.



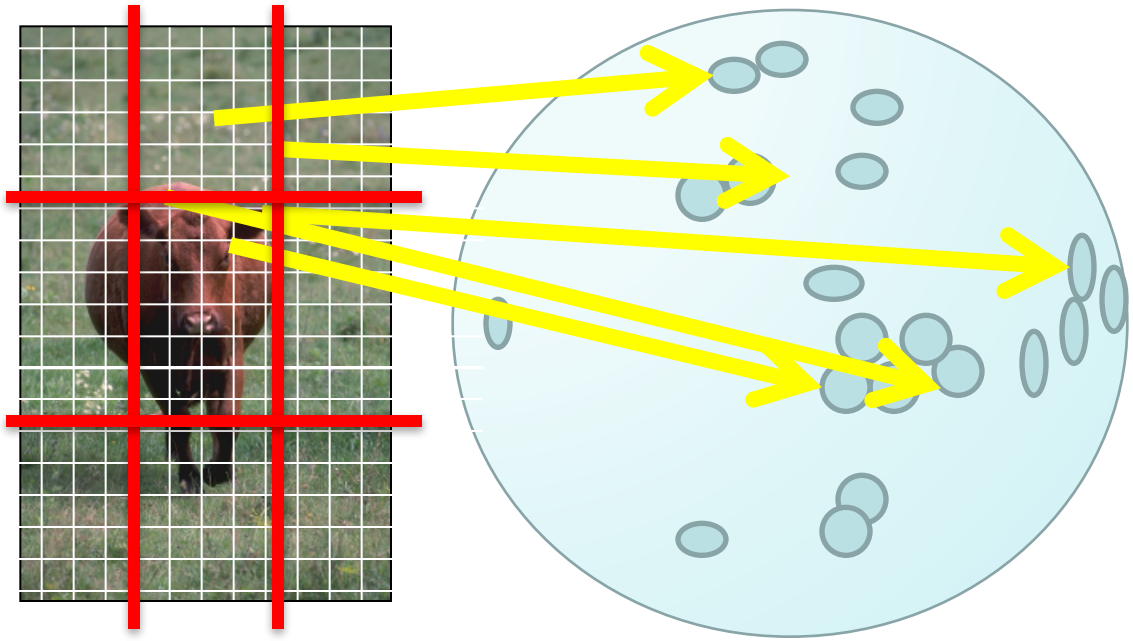
Tinne Tuytelaers discusses dense saliency.

Gather region by region

A spatial pyramid gathers location-dependent evidence

3x3 two-layer pyramid is good.

More-layered reduction is better.

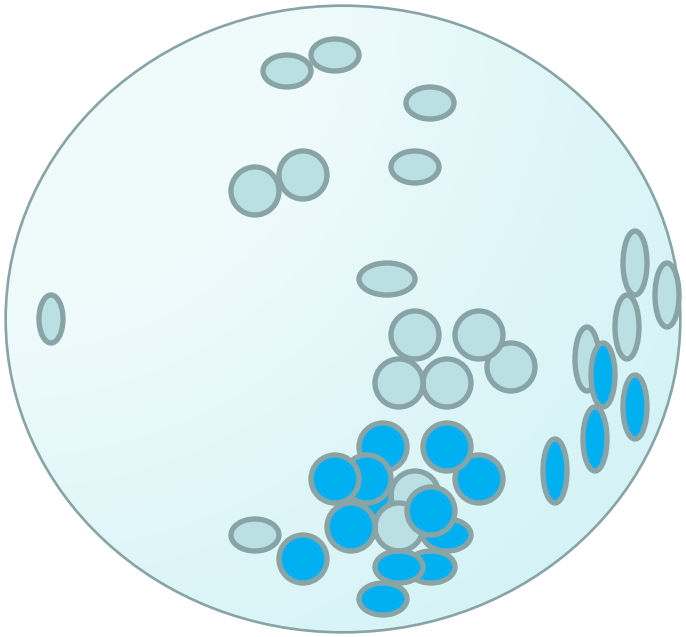


Sample many images

Sample the images in the world: the learning set.

Learn all relevant variability between types.

Learn all irrelevant variations not covered by invariance.



Form a dictionary of visual words

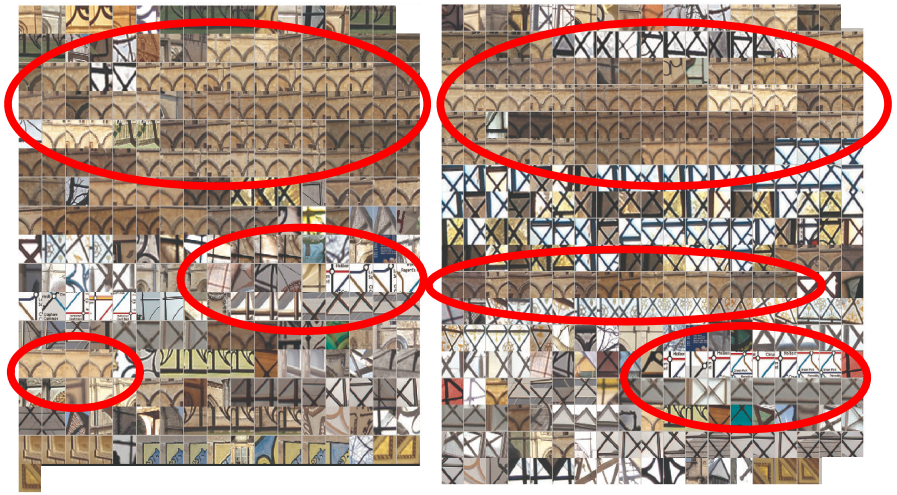
Form regions in feature space: 4,000 visual words

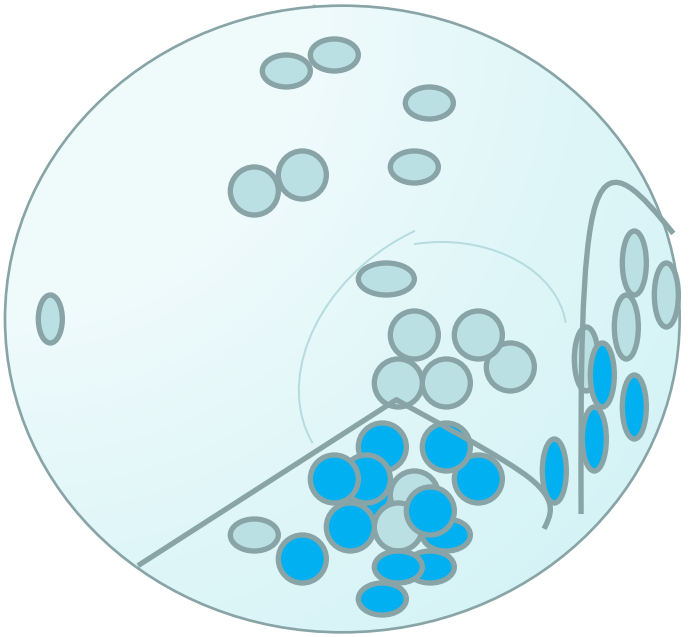
More words needed for fine detail discrimination.

Words cover several types of patches

Collections of words are still discriminative.

Similar patches may emerge distant.



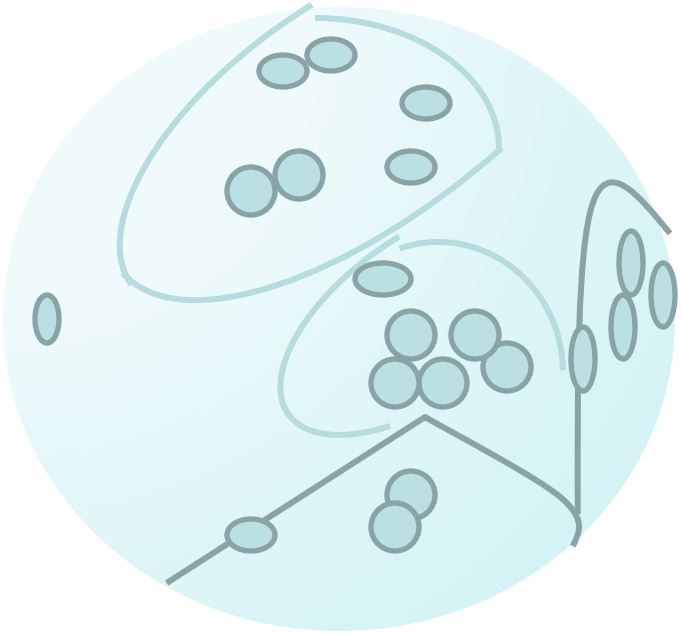


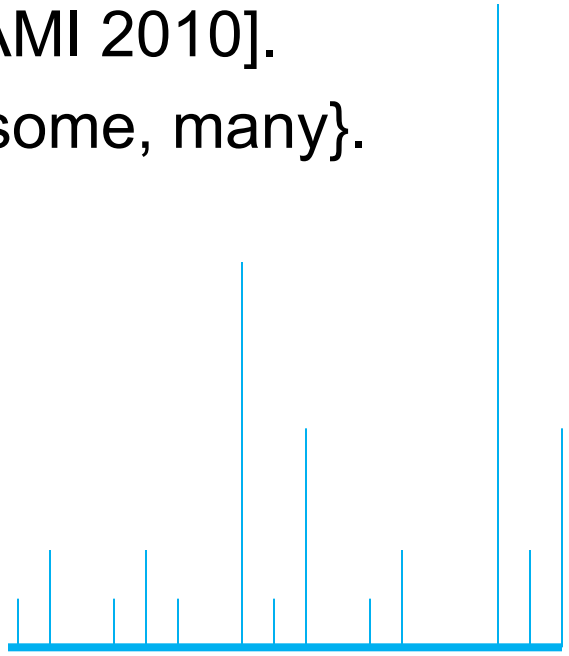
Encode patches into words

Word occurrences per training image: area covered.

Soft assign to code words [Van Gemert PAMI 2010].

Make a word histogram: {none, rare, few, some, many}.



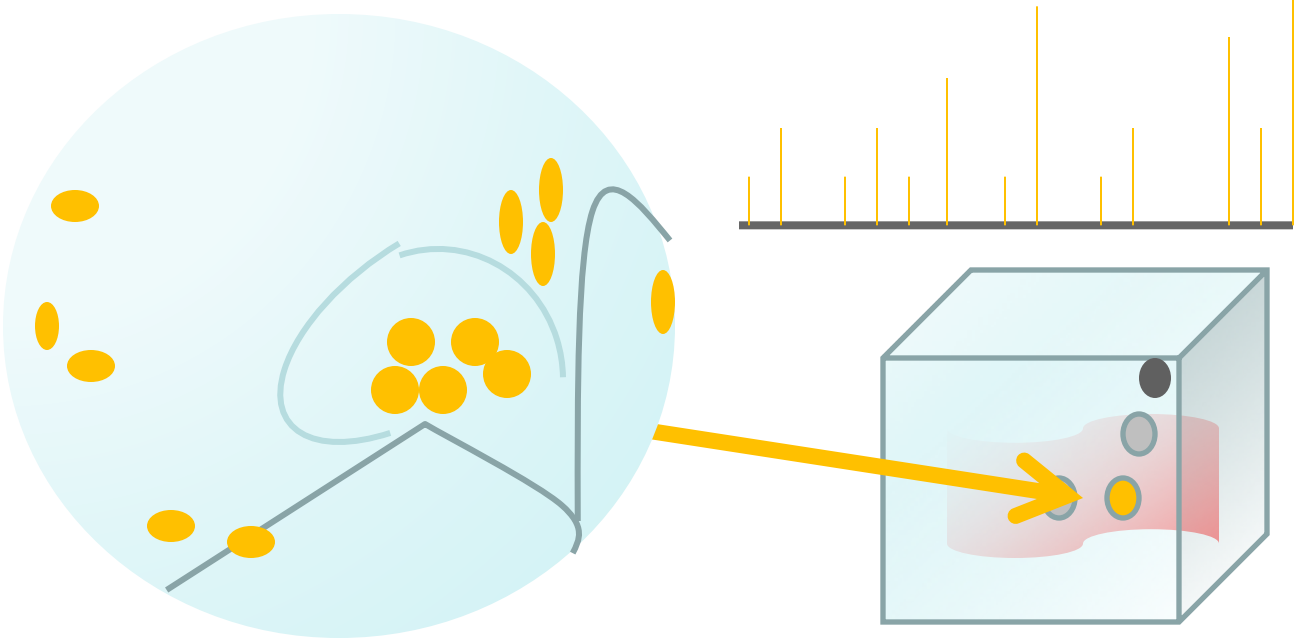


Classify histogram of words

Learn the word-count histogram **V**j = {t1, t2, … tN} per type.

Similarity between **V**q and each vector **V**j in the dataset by:

normalized scalar product = **V**q . **V**j / ||**V**q||.||**V**t||



BoW

Features large dimensionality for omni-potency.

powerful normalization & invariance.

Multi-stage encode – classify process to separate:

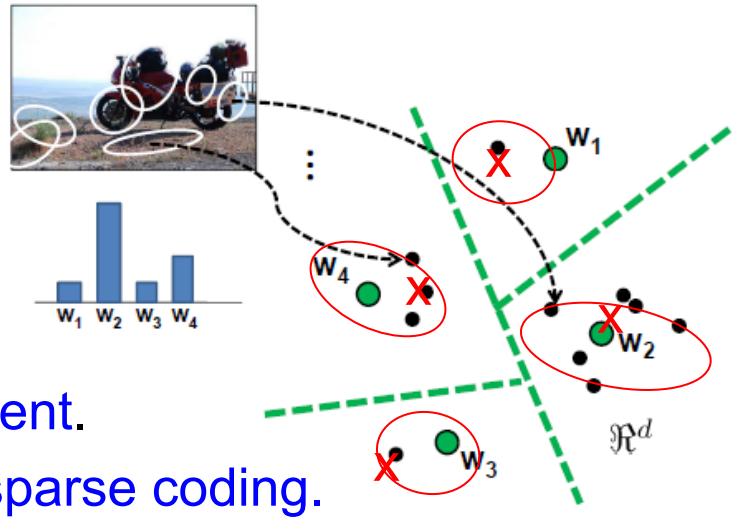
the variabilities of the world.

from the invariability of the class.

5. Encoders

We consider the external and internal structure of a code word, as well as the local structure around the code word in feature space. What is the external structure of a code? We discuss shallow and deep encoders. What does the internal structure of a code reveal? Knowledge of the distribution within the words delivers two new representations: Fisher and VLAD.

External & Internal Structure



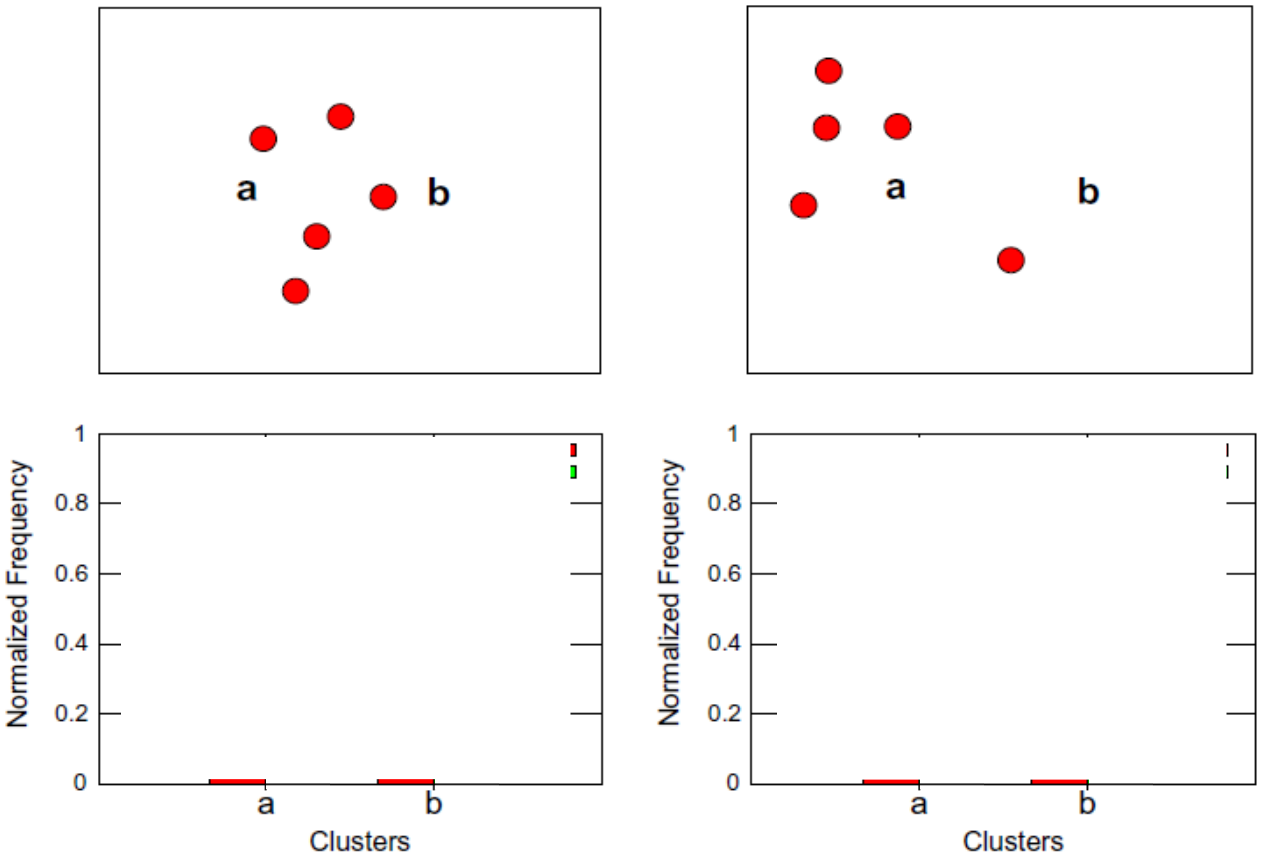
External structure

1. Actual words boundaries pretty arbitrary.
2. Use multiple neighbor codes: soft assignment.
3. Use over complete but sparse dictionary: sparse coding.
4. Deep learning by autoencoder.

Internal structure

1. Internal structure of words ignored by (hard, soft, sparse) counting.
2. Use other statistics > Mean subtract: VLAD / GMM: Fisher

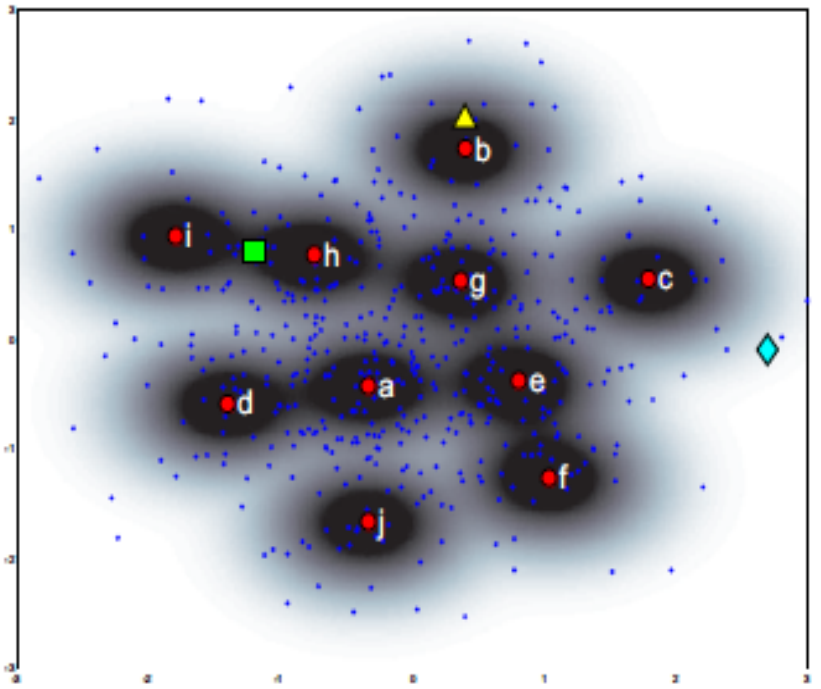
Quiz: How to count codes? (the hard way)



External: soft word assignment

Assign to more than 1 word by weights

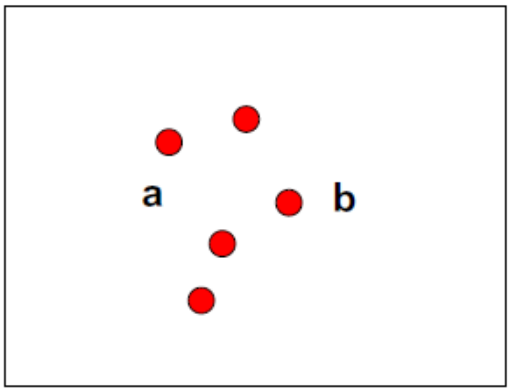
Code word uncertainty:

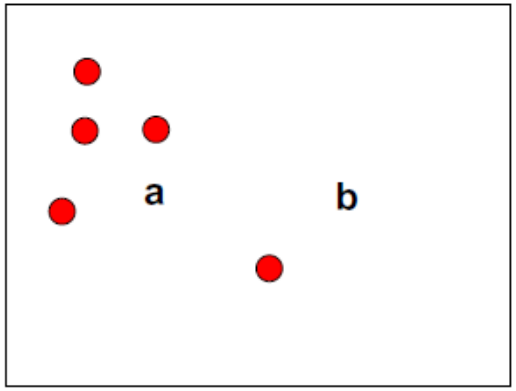


$$\mathrm{UNC}(w) = \frac{1}{n}\sum_{i=1}^{n}\frac{K_\sigma\left(D\left(w,r_i\right)\right)}{\sum_{j=1}^{|V|}K_\sigma\left(D(v_j,r_i)\right)},$$

Van Gemert PAMI 2010

Quiz: How to count codes? (the soft way)





External: sparse coding

Originally developed to explain early visual processing in the brain by means of edge encodings

Objective: Given a set of input data vectors ($x^{(1)}$, $x^{(2)}$, …, $x^{(m)}$) learn a dictionary of bases (ϕ_1, ϕ_2, …, ϕ_k) such that:

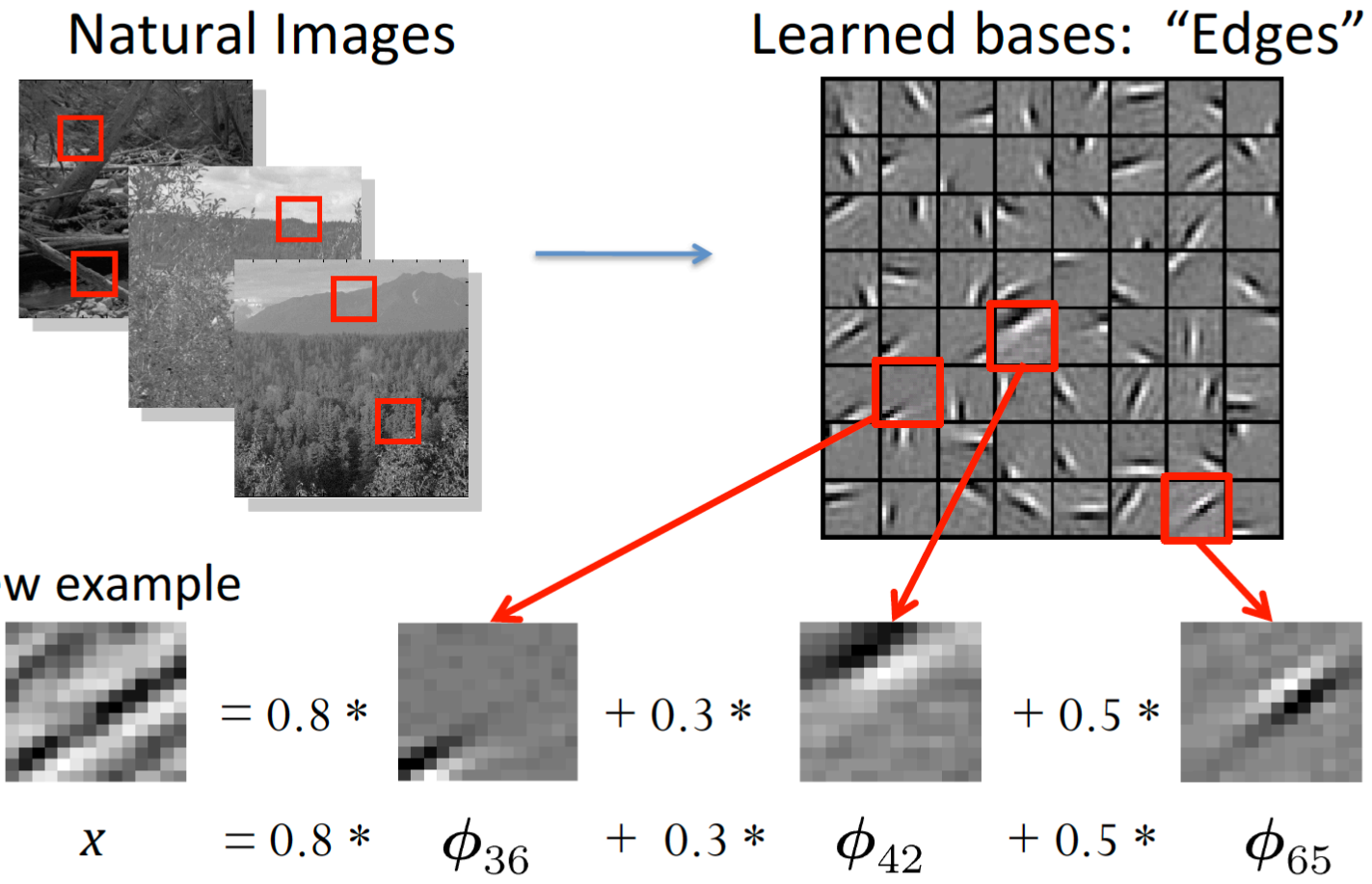
$$x \approx \sum_{j=1}^{k} a_j \phi_j$$

Sparse: enforce most to be zero

Each data vector is encoded as linear combination of bases

Olshausen & Field, Nature 1996

External: sparse coding



[0, 0, … **0.8**, …, **0.3**, …, **0.5**, …] = coefficients (feature representation)

Sparse coding: training

Input: Images $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ (each in $\mathrm{R}^{n \times n}$)

$$\min_{a,\phi} \sum_{i=1}^{m} \left(\left\| x^{(i)} - \sum_{j=1}^{k} a_j^{(i)} \phi_j \right\|^2 + \lambda \sum_{j=1}^{k} |a_j^{(i)}| \right)$$

Reconstruction error

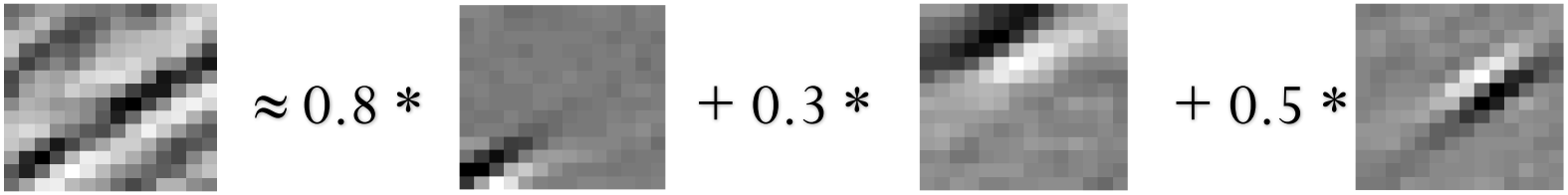
Sparsity penalty

Alternately minimize with respect to ϕ_i's (easy) and *a*'s (harder).

Sparse coding: testing

Input: Novel image x (in $\mathrm{R}^{n \times n}$) and previously learned ϕ_i's.
Output: Representation $[a_1, a_2, \ldots, a_k]$ of image x.

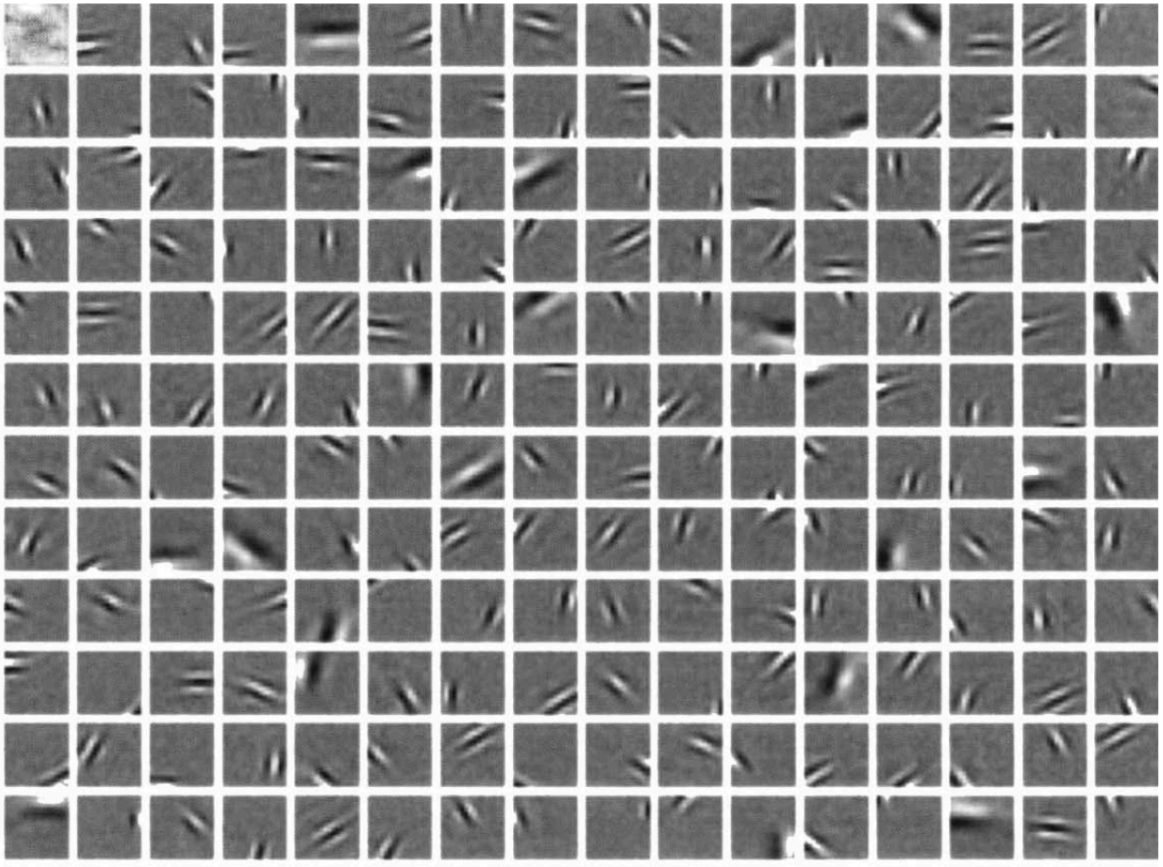
$$\min_a \left\| x - \sum_{j=1}^{k} a_j \phi_j \right\|^2 + \lambda \sum_{j=1}^{k} |a_j|$$



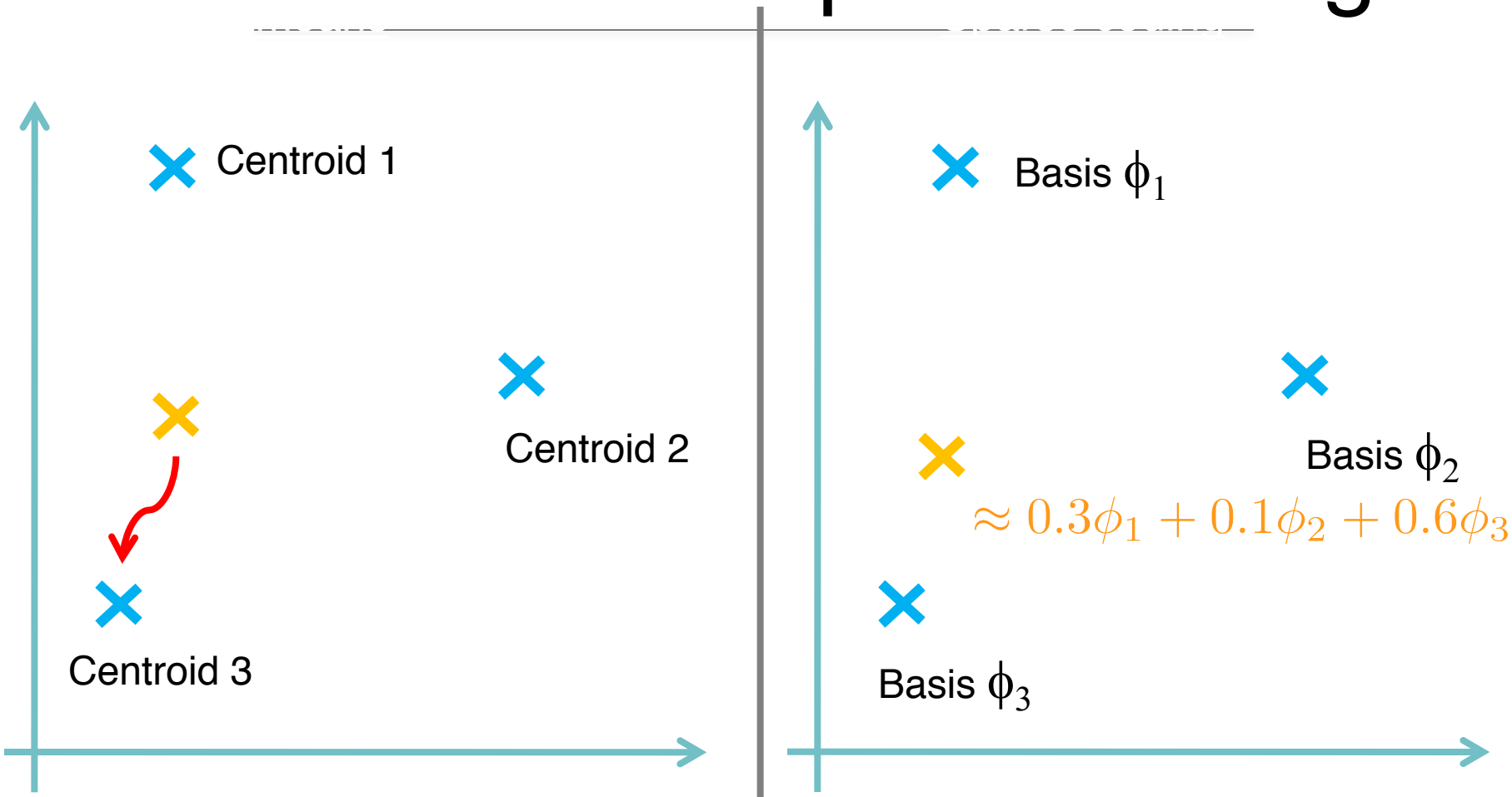
Represent as: [0, 0, …, 0, **0.8**, 0, …, 0, **0.3**, 0, …, 0, **0.5**, …]

Results

Training 192 basis functions on 16x16 image patches from natural scene images



K-means vs. sparse coding



Represent as: $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

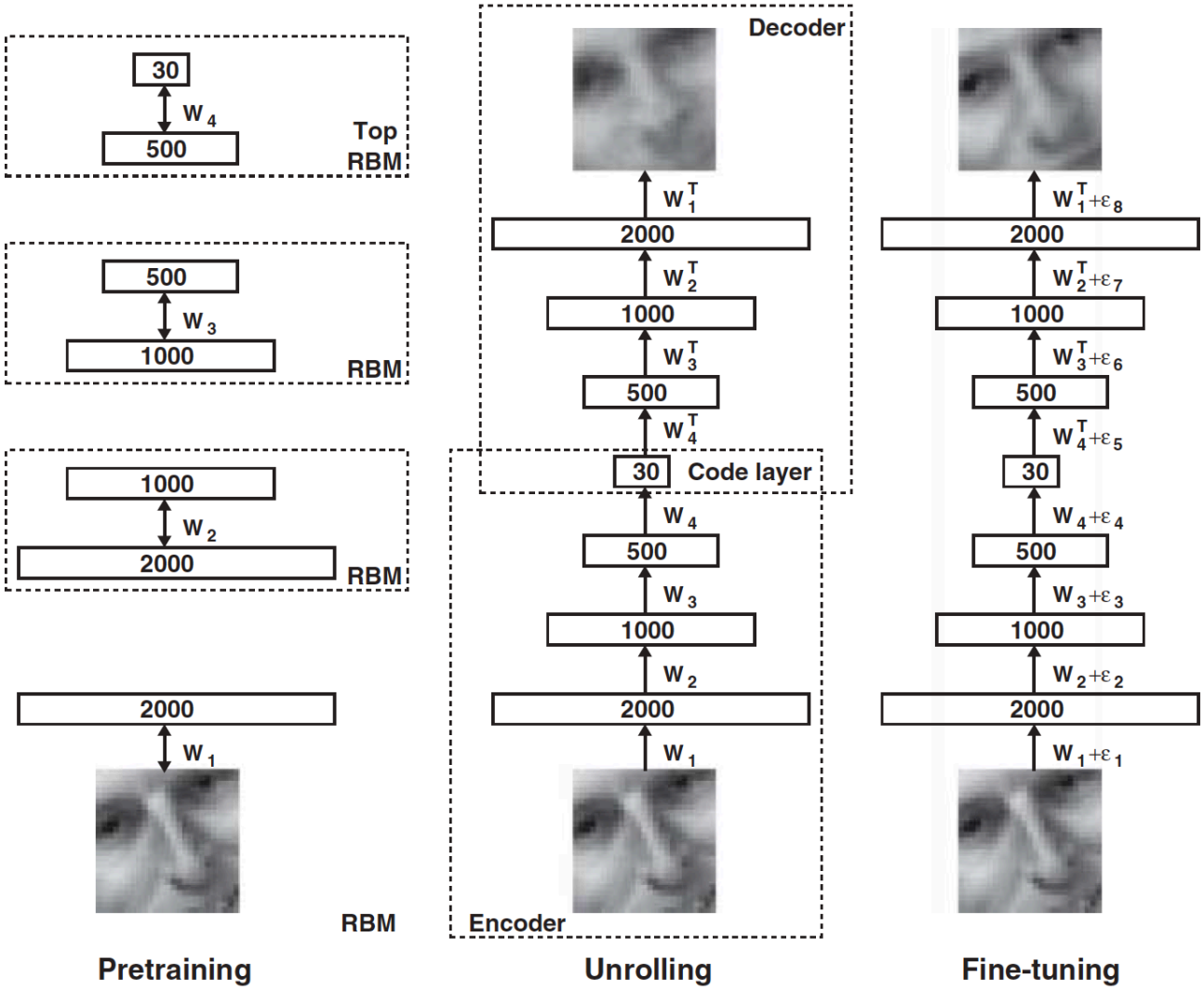
Represent as: $\begin{bmatrix} 0.3 \\ 0.1 \\ 0.6 \end{bmatrix}$

K-means vs. sparse coding

Rule of thumb:

Whenever using k-means to get a dictionary, if you replace it with sparse coding it will often work better.

External: Autoencoder

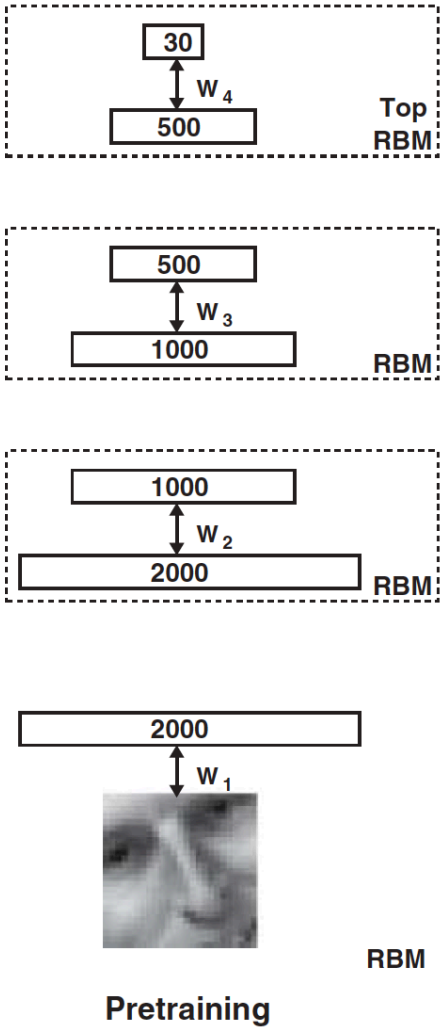


Hinton & Salakhutdinov, Science 2006

Autoencoder

Introduction of pretraining

Layer-by-layer learning

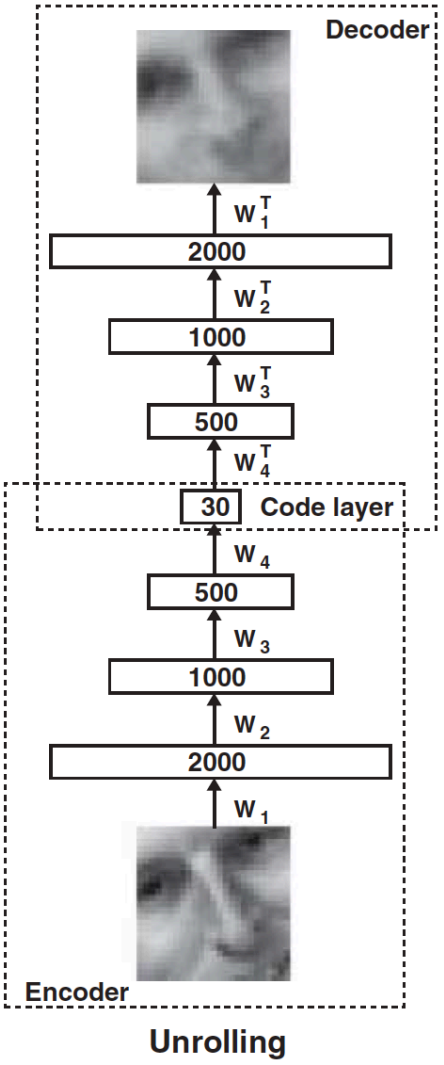


Autoencoder

Introduction of pretraining

Layer-by-layer learning

Unrol into encoder and decoder



Unrolling

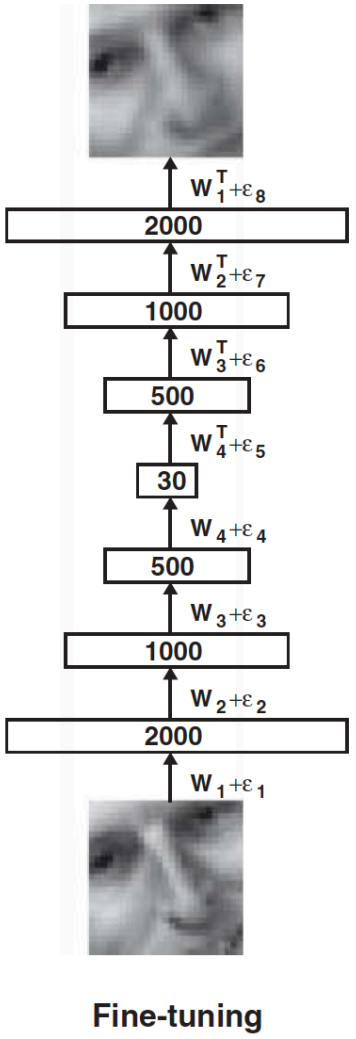
Autoencoder

Introduction of pretraining

Layer-by-layer learning

Unrol into encoder and decoder

Finetune entire network



Fine-tuning

Some results

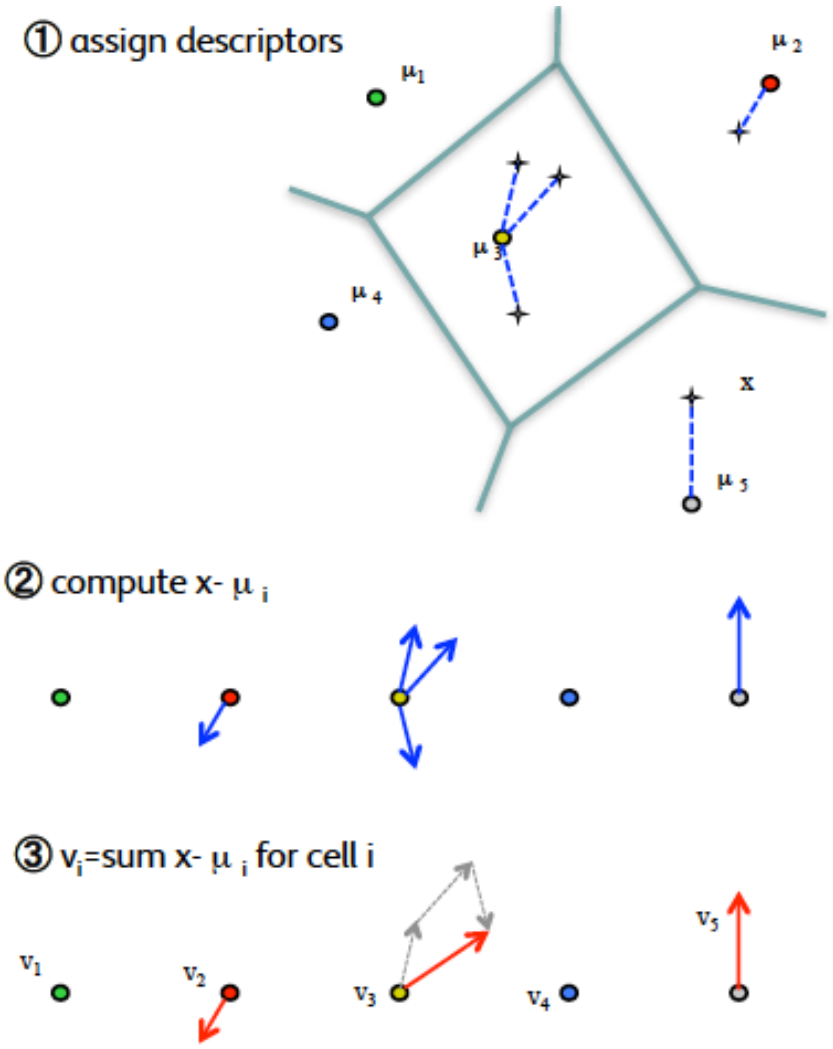
Autoencoder to extract 30d codes for Olivetti face images

Input

Autoencoder

PCA

Internal: VLAD



Jegou CVPR 2010

Hard assignment to nearest word.

Subtract learned mean for differential coding efficiency.

Rather than count, sum all code residues within a word.

Concatenate all word sums and l2-normalize.

Memory efficient coding.

Internal: VLAD

Red = residual of **v** for 16 words of 128 dimensions each.

Jegou CVPR 2010

Internal: Fisher vector

Score *G* of sample *X* given GMM-likelihood *u* with param λ:

$$G_\lambda^X = \nabla_\lambda \log u_\lambda(X)$$

Derivative reveals differential GMM-parameters for the sample.

Form the Fisher matrix to measure …

$$F_\lambda = E_{x \sim u_\lambda}\left[\nabla_\lambda \log u_\lambda(x) \nabla_\lambda \log u_\lambda(x)'\right]$$

… similarity between samples by the (inversed) Fisher kernel:

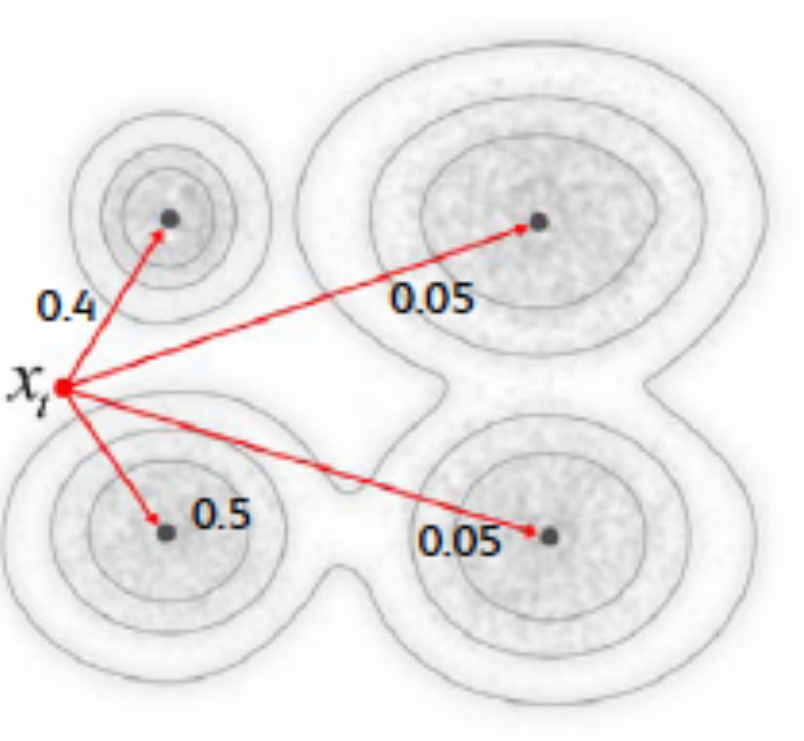
$$K(X,Y) = G_\lambda^{X\prime} F_\lambda^{-1} G_\lambda^Y$$

$$\mathcal{G}_\lambda^X = L_\lambda G_\lambda^X \qquad G_\lambda^X = \frac{1}{T}\sum_{t=1}^{T} \nabla_\lambda \log u_\lambda(x_t)$$

This transform spreads out all encodings for better distinction.

Perronnin CVPR 2007, ECCV 2010

Internal: Fisher vector



Compare N words to 2DN for D feature dimensions.

Perronnin CVPR 2007, ECCV 2010

Internal: Fisher vector

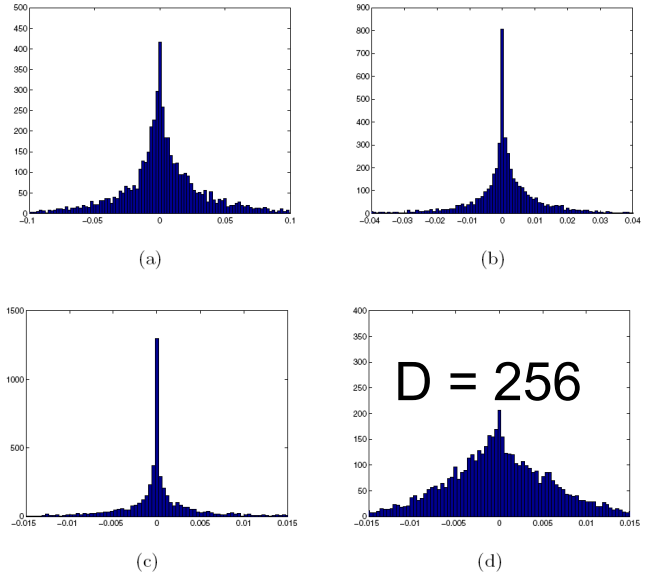
Fisher vector with diagonal covariance matrix:

5% better than BoW, smaller D of codebooks needed.

Improved by using the L2 – norm.

Improved by using a power – norm.

Improved by spatial pyramids.



Perronnin CVPR 2007, ECCV 2010

Quiz: What is the relation between soft, sparse and Fisher?

Write a paper.

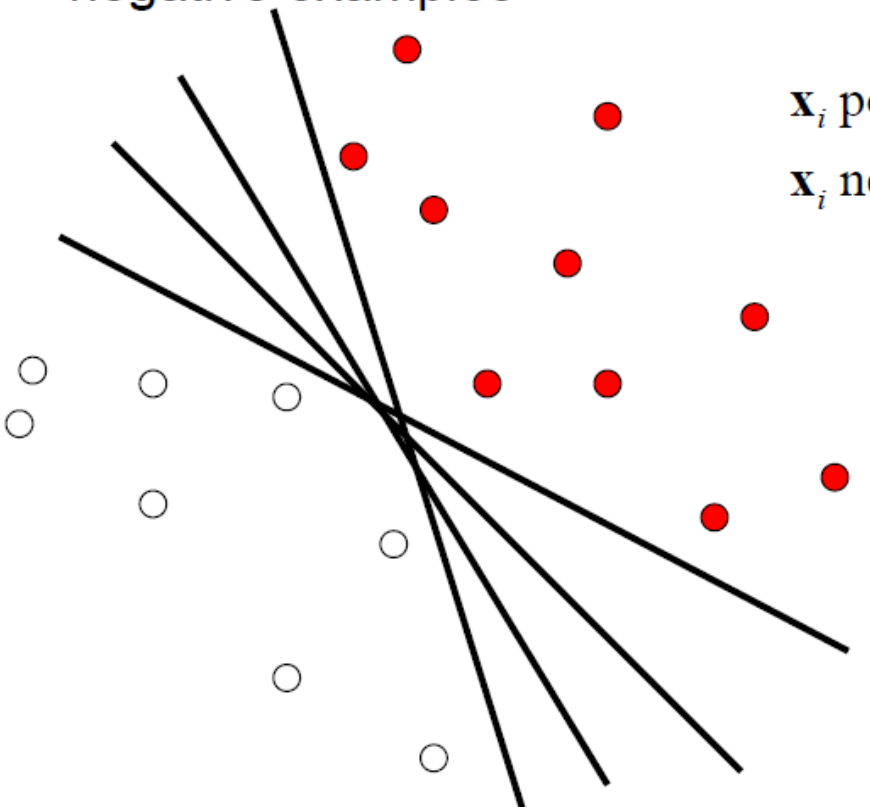
6. Support Vector Machine

The support vector machine separates an *n*-dimensional feature space into a class of interest and a class of disinterest by means of a hyperplane. A hyperplane is considered optimal when the distance to the closest training examples is maximized for both classes. The examples determining this margin are called the support vectors. For nonlinear margins, the SVM exploits the kernel trick. It maps the distance between feature vectors into a higher dimensional space in which the hyperplane separator and its support vectors are obtained as easy as in the linear case. Once the support vectors are known, it is straightforward to define a decision function for an unseen test sample.

Vapnik, 1995

Quiz: What linear classifier is best?

Find linear function (*hyperplane*) to separate positive and negative examples



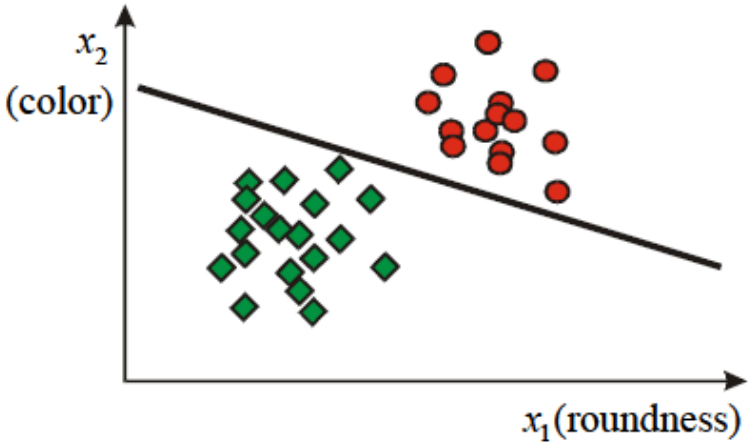
$\mathbf{x}_i$ positive: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

$\mathbf{x}_i$ negative: $\mathbf{x}_i \cdot \mathbf{w} + b < 0$

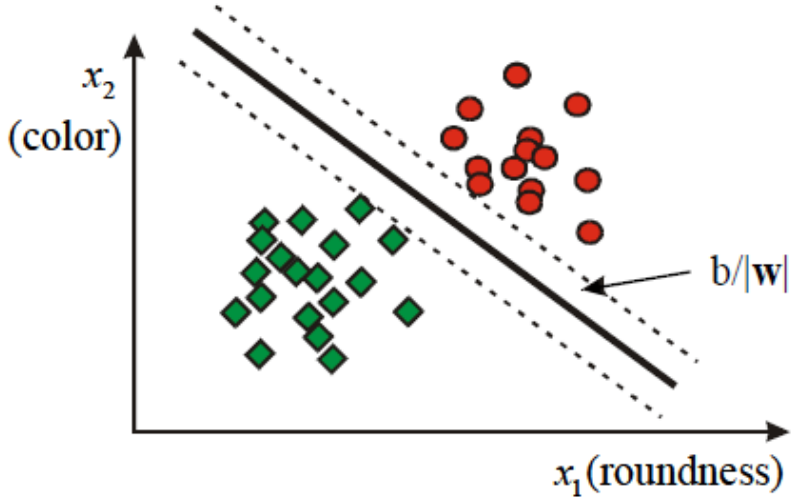
Which hyperplane is best?

Linear classifiers - margin

Generalization is not good in this case:



Better if a margin is introduced:



Training a linear SVM

To find the maximum margin separator, we have to solve the following optimization problem:

$$\mathbf{w}.\mathbf{x}^c + b > +1 \quad for\ positive\ cases$$

$$\mathbf{w}.\mathbf{x}^c + b < -1 \quad for\ negative\ cases$$

$$and \quad \| \mathbf{w} \|^2 \ is\ as\ small\ as\ possible$$

Convex problem. Solved by quadratic programming.

Software available: LIBSVM, LIBLINEAR

Quadratic w.r.t. the number of training samples (ouch!)

For big datasets approximations mandatory

Testing a linear SVM

The separator is defined as the set of points for which:

$$\mathbf{w}.\mathbf{x} + b = 0$$

$$so\ if\ \ \mathbf{w}.\mathbf{x}^c + b > 0\ \ say\ its\ a\ positive\ case$$

$$and\ if\ \ \mathbf{w}.\mathbf{x}^c + b < 0\ \ say\ its\ a\ negative\ case$$

L2 Normalization

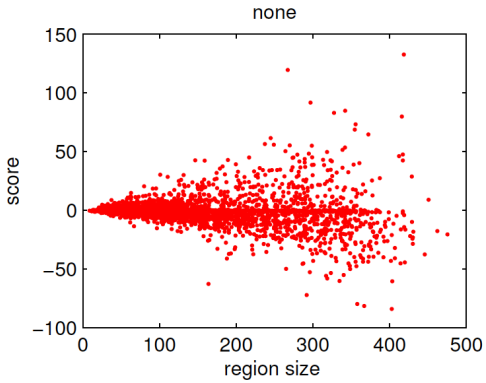
Linear classifier for object and scene classification prefers L2 normalization [Vedaldi ICCV09]

$$\|\mathbf{x}\| = \sqrt{\sum_{k=1}^{n} |x_k|^2},$$

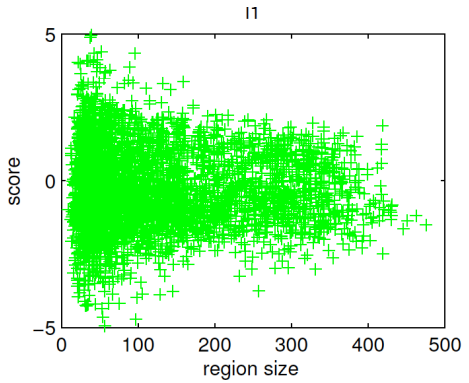
Important for Fisher vector

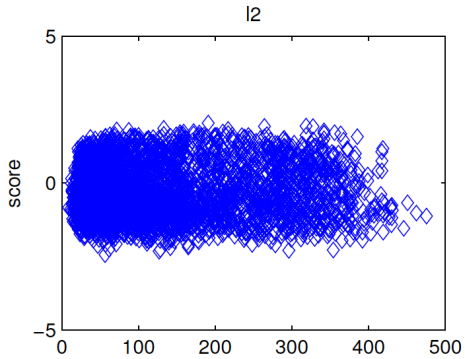
Acts as scale invariant

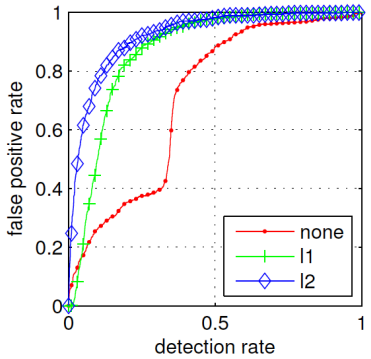
Large object bias



Small object bias

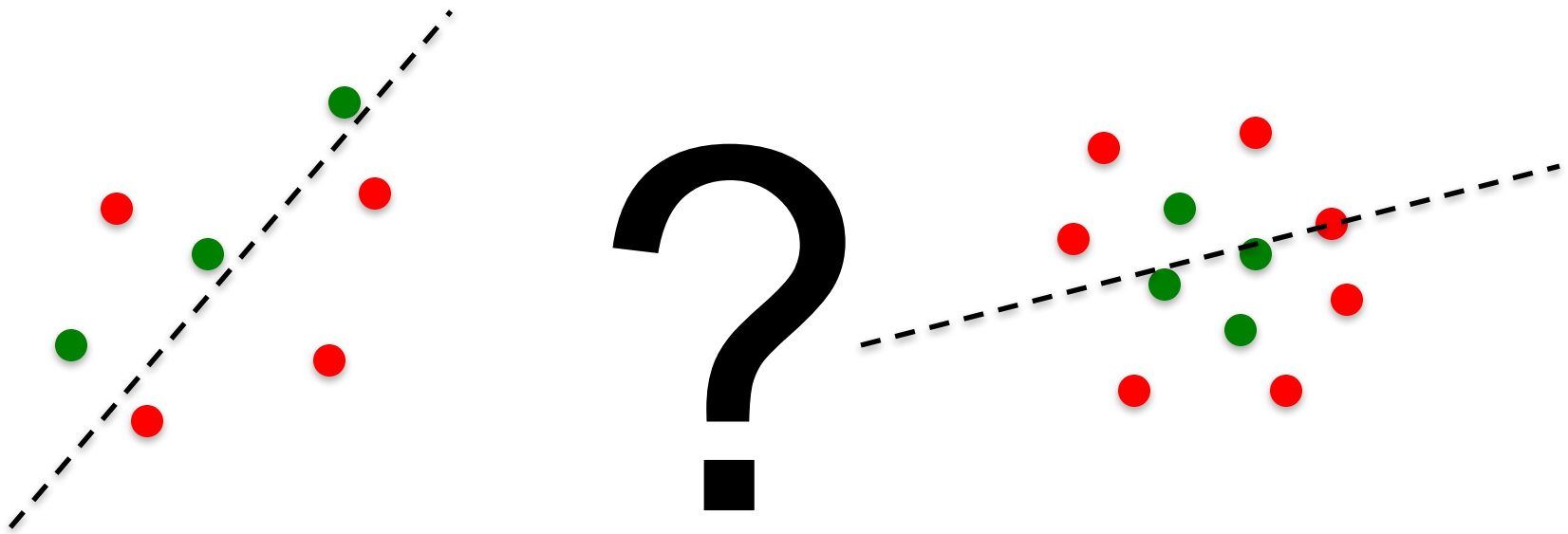






No scale bias

Quiz: What if data is not linearly separable?



Solutions for non separable data

1. Slack variables

2. Feature transformation

1. Introducing slack variables

Slack variables are constrained to be non-negative. When they are greater than zero they allow us to cheat by putting the plane closer to the datapoint than the margin. So we need to **minimize the amount of cheating**. This means we have to pick a value for lambda

$$\mathbf{w}.\mathbf{x}^c + b \geq +1 - \xi^c \quad for\ positive\ cases$$

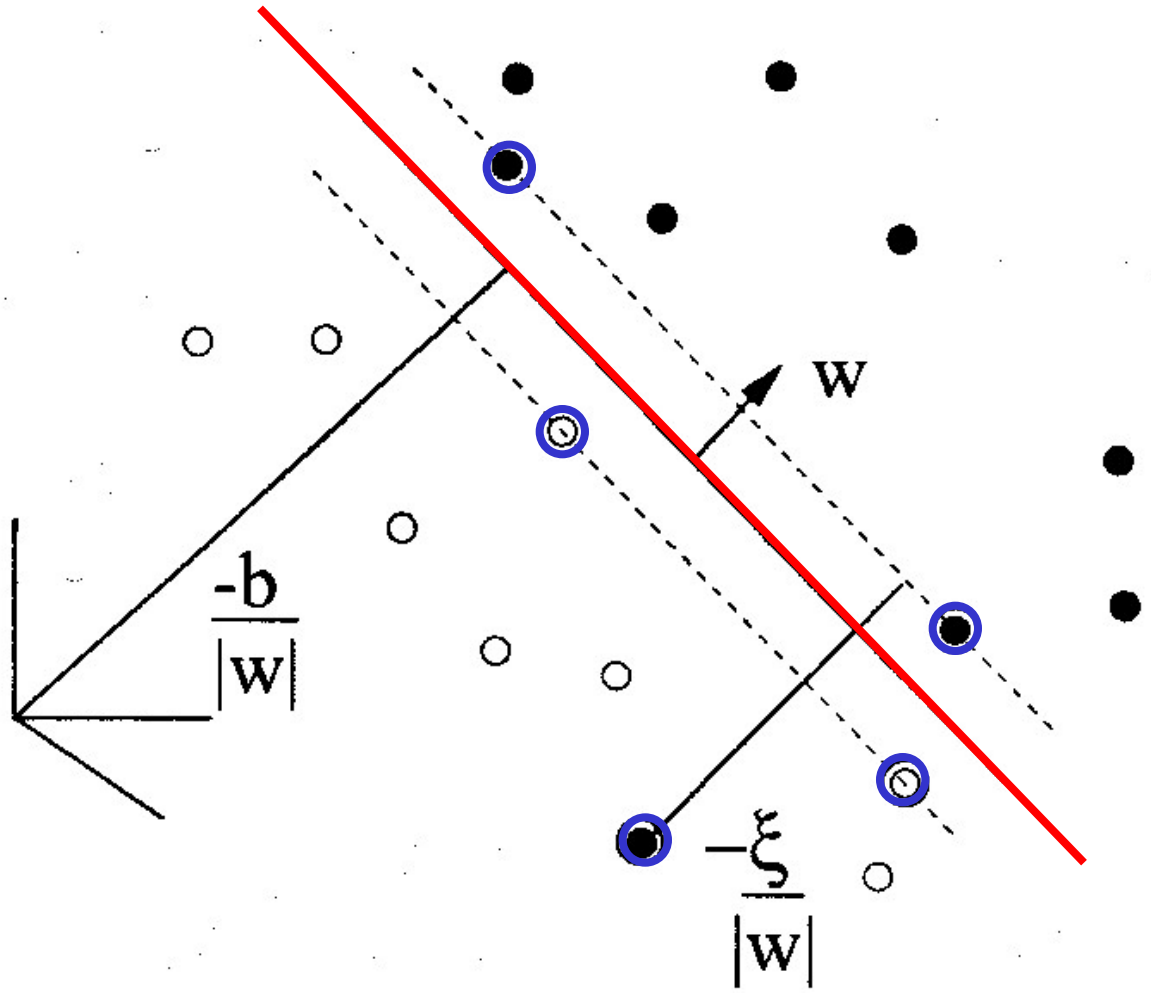
$$\mathbf{w}.\mathbf{x}^c + b \leq -1 + \xi^c \quad for\ negative\ cases$$

$$with\ \ \xi^c \geq 0 \quad for\ all\ c$$

$$and\ \ \frac{\|\mathbf{w}\|^2}{2} + \lambda \sum_c \xi^c \quad as\ small\ as\ possible$$

Slide credit: Geoff Hinton

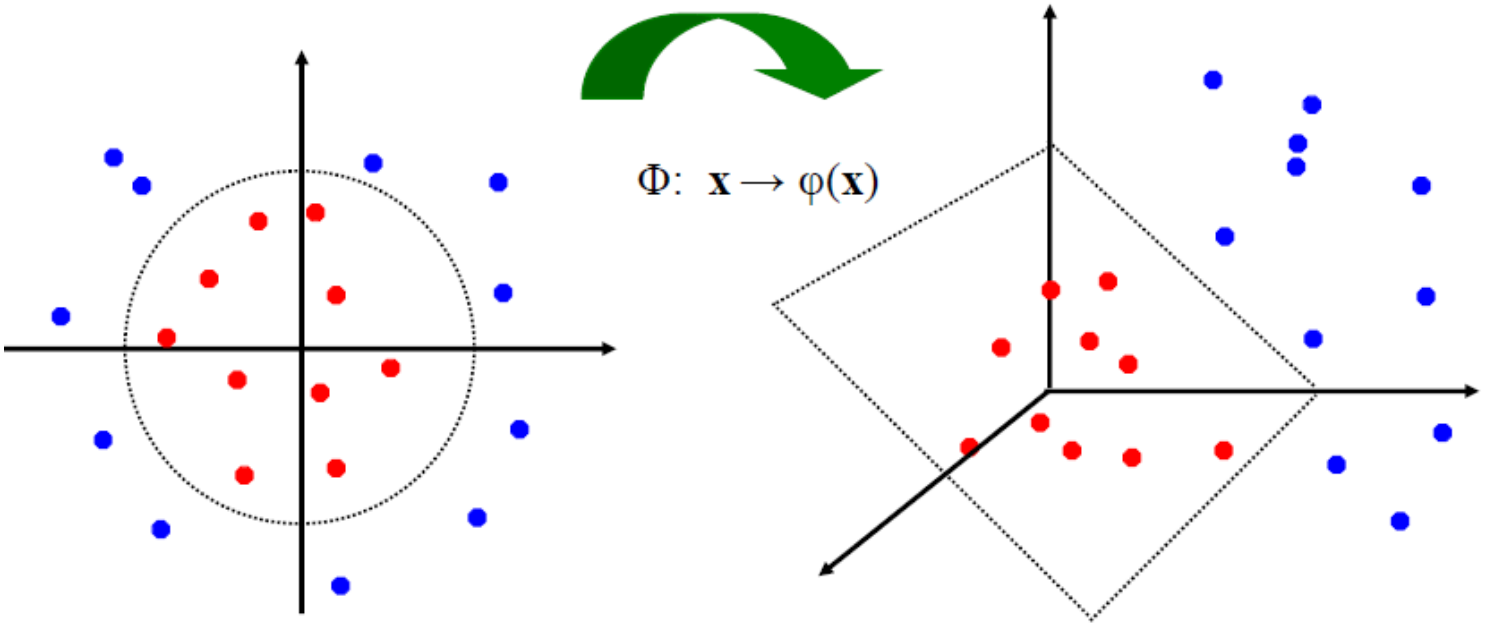
Separator with slack variable



Slide credit: Geoff Hinton

2. Feature transformations

Transform the feature space in order to achieve linear separability after the transformation.



Slide credit: Andrew Moore

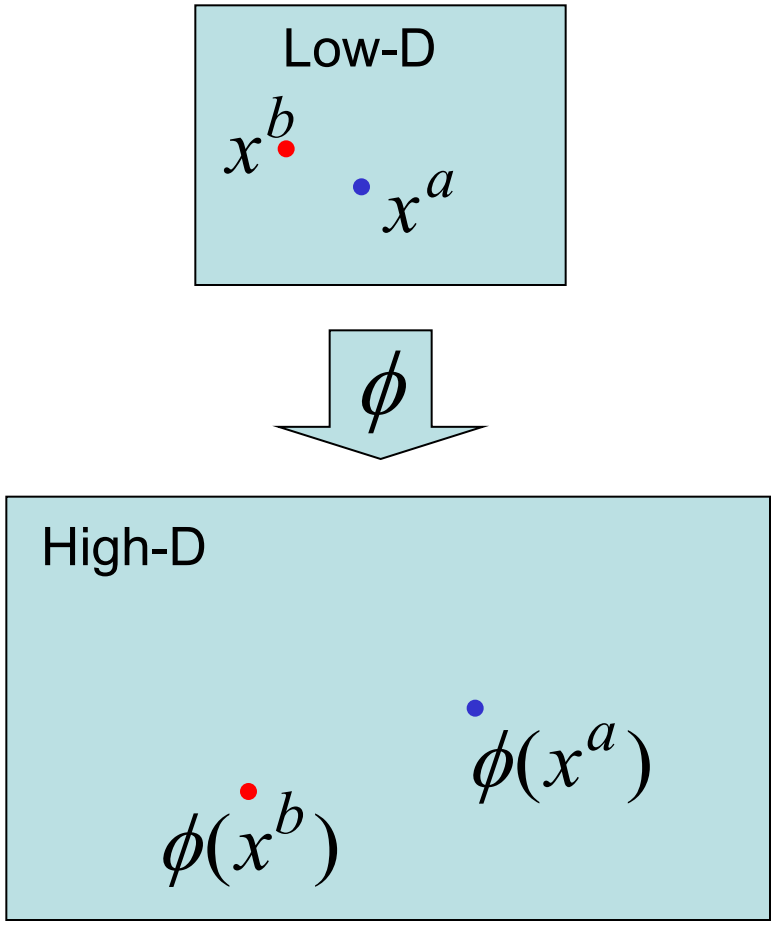
The kernel trick

For many mappings from a low-D space to a high-D space, there is a simple operation on two vectors in the low-D space that can be used to compute the scalar product of their two images in the high-D space.

$$K(x^a, x^b) = \phi(x^a).\phi(x^b)$$

Letting the kernel do the work

doing the scalar product in the obvious way



Slide credit: Geoff Hinton

The classification rule

The final classification rule is quite simple:

$$bias + \sum_{s \varepsilon SV} w_s K(x^{test}, x^s) > 0$$

The set of support vectors

All the cleverness goes into selecting the support vectors that maximize the margin and computing the weight to use on each support vector.

.

Slide credit: Geoff Hinton

Popular kernels for computer vision

Histogram intersection kernel:

$$I(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i))$$

Generalized Gaussian kernel:

$$K(h_1, h_2) = \exp\left(-\frac{1}{A} D(h_1, h_2)^2 \right)$$

D can be Euclidean distance, χ^2 distance, Earth Mover's Distance, etc.

$$D(h_1, h_2) = \sum_{i=1}^{N} \frac{\left(h_1(i) - h_2(i)\right)^2}{h_1(i) + h_2(i)}$$

Slide credit: Cordelia Schmid

Linear vs non-linear kernels

	Linear	Non-linear
Training speed	Very fast	Very slow
Training scalability	Very high	Low
Testing speed	Very fast	Very slow
Test accuracy	Lower	Higher

Slide credit: Jianxin Wu

Nonlinear kernel speedups

Many have proposed speedups for nonlinear kernels.

Exploiting two basic properties:

Additivity

$$K(p, q) = K(p_1, q_1) + \cdots + K(p_D, q_D)$$

Homogeneity

$$K(c \cdot p, c \cdot q) = c^{\gamma} K(p, q)$$

Fast HIK as fast as linear kernel exploiting additivity

Feature maps for all additive homogeneous kernels.

Maji et al. PAMI 2013

Vedaldi et al. PAMI 2012

Gavves, CVPR 2012

Selecting and weighting dimensions

For additive kernels *all dimensions are equal*

$$K_{\mathbf{X},\,\mathbf{Y}} = 1 \cdot k^1_{\mathbf{X},\,\mathbf{Y}} + \cdots + 1 \cdot k^D_{\mathbf{X},\,\mathbf{Y}}$$

We introduce scaling factor *c*

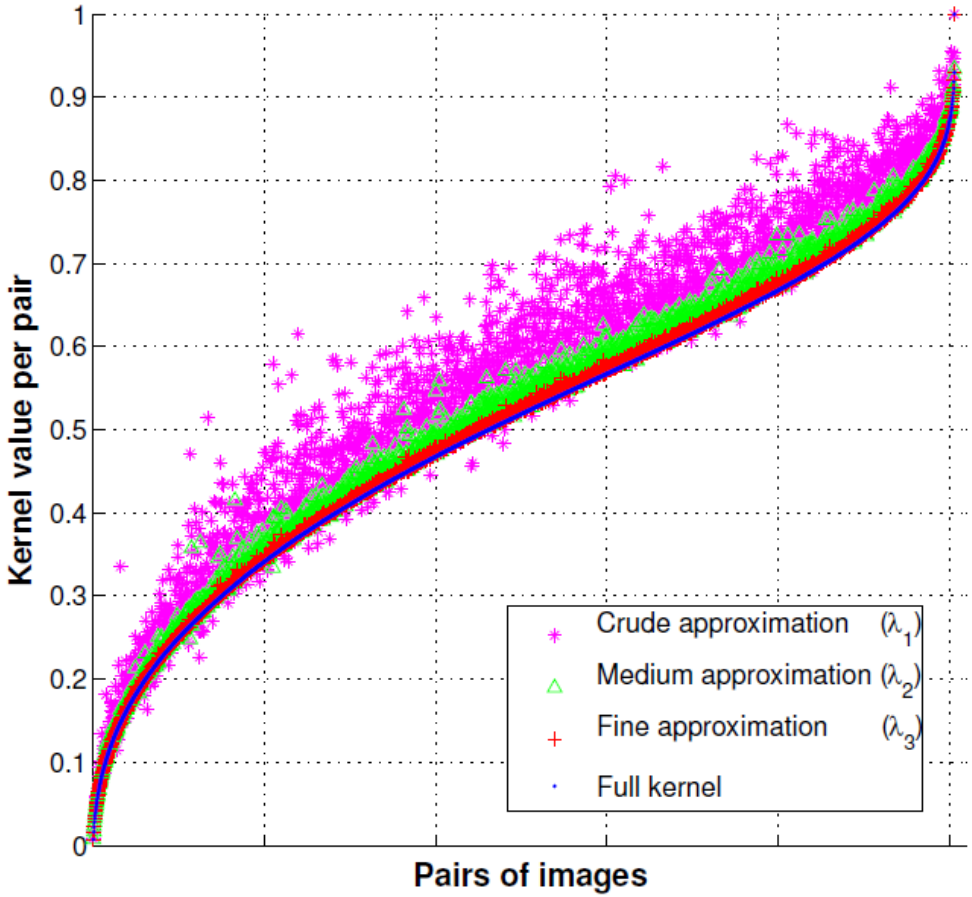
$$K_{\mathbf{X},\,\mathbf{Y}} = c_1 \cdot k^1_{\mathbf{X},\,\mathbf{Y}} + \cdots + c_D \cdot k^D_{\mathbf{X},\,\mathbf{Y}} = \mathbf{c}^T \cdot \mathbf{k}_{\mathbf{X},\,\mathbf{Y}}$$

Kernel reduction as convex optimization problem

$$\arg\min_{\mathbf{c}} \left\| \mathbf{K}_{\mathbf{X},\,\mathbf{Y}} - \mathbf{c}^T \cdot \mathbf{k}_{\mathbf{X},\,\mathbf{Y}} \right\|^2 + \lambda \left\| \mathbf{c} \right\|_{\ell 1}$$

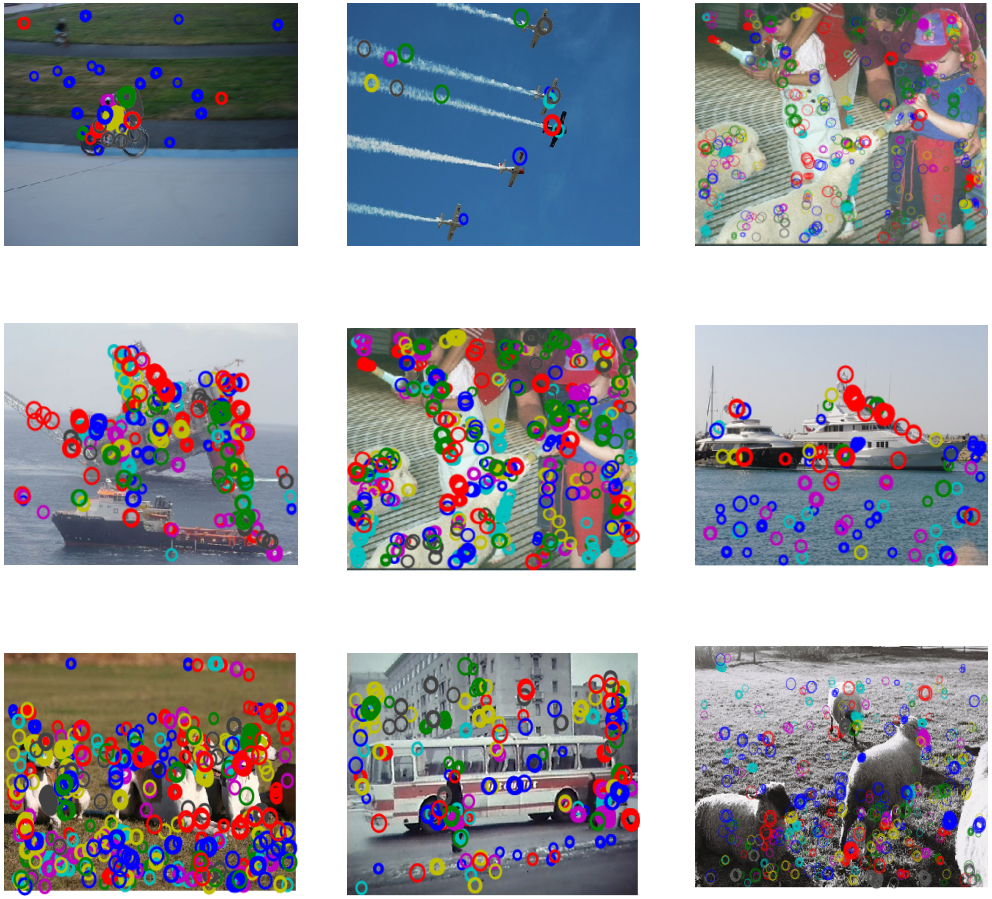
Sparse coding of the kernel

Similar accuracy with a 45-85% smaller size.



Equally accurate and 10x faster as PCA codebook reduction.
Applies also to Fisher vectors.

Selected kernel dimensions



Note: descriptors originally dense sampled

Performance

Support Vector Machines work very well in practice.

- The user must choose the kernel function and its parameters, but the rest is automatic.
- The test performance is very good.

They can be expensive in time and space for big datasets

- The computation of the maximum-margin hyper-plane depends on the square of the number of training cases.
- We need to store all the support vectors.
- Exploit kernel additivity and homogenity for speedup

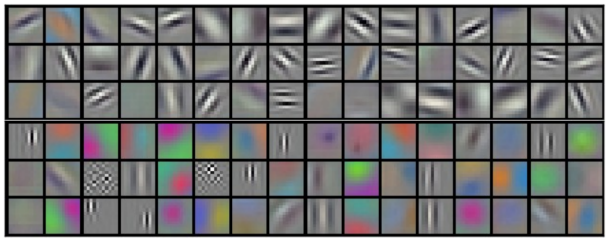
SVM's are very good if you have no idea about what structure to impose on the task.

7. Finale: BoW versus CNN

Bag of words is a multi-stage code & classify process aimed at separating variability of feature values from invariant labels.

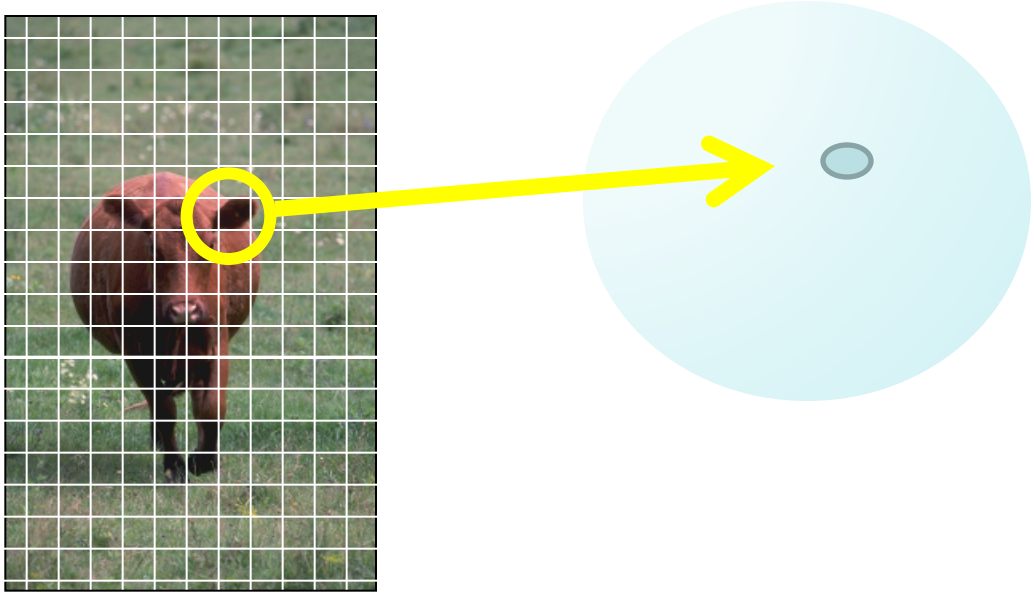
CNN is a multi-stage code & classify process aimed at separating variability of images from invariant labels.

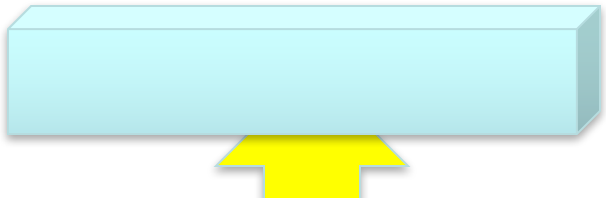
Capture the pattern in each patch



Measure the pattern in a patch with abundant features.

More patches is better. More features is better. Normalized.

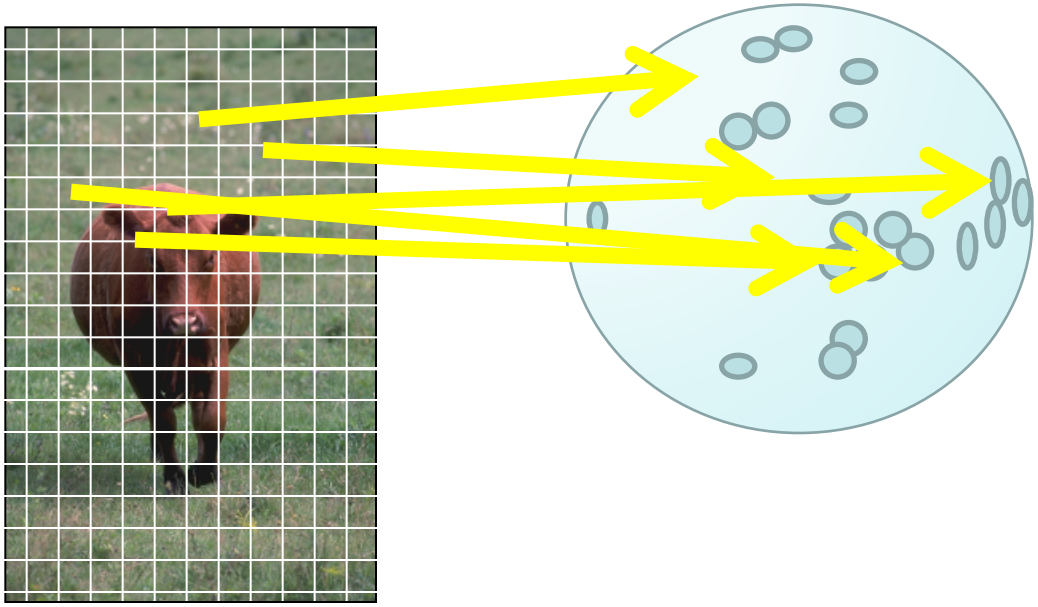


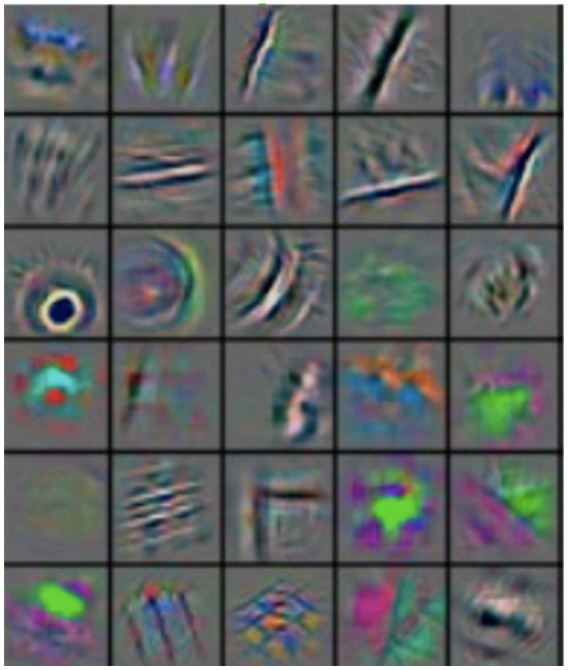


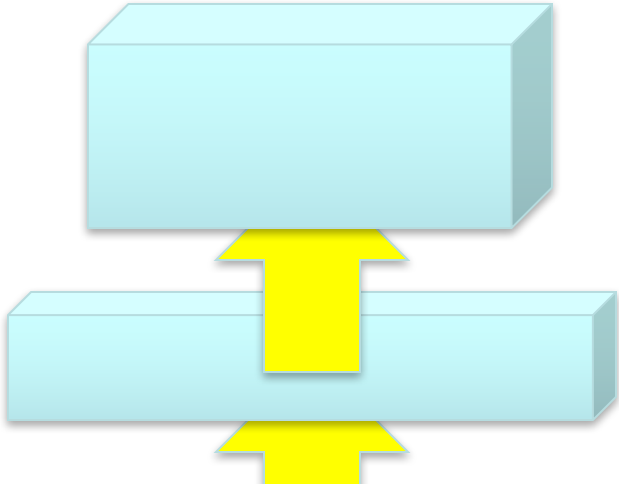
Sample many patches

Sample the patches in the image.

Salience is good, dense better. us





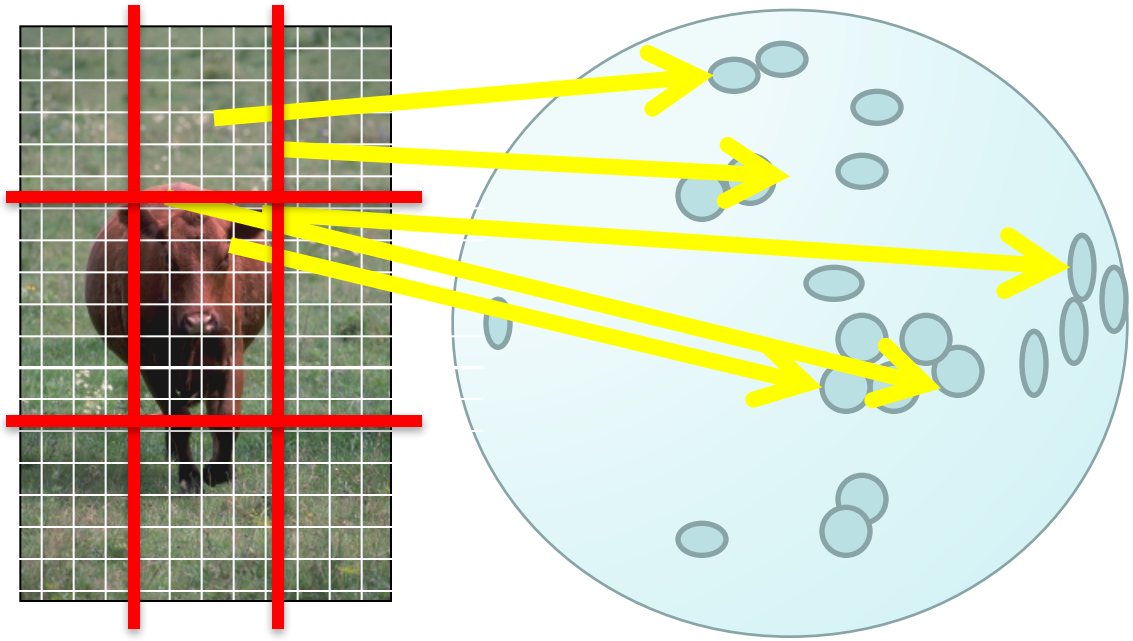


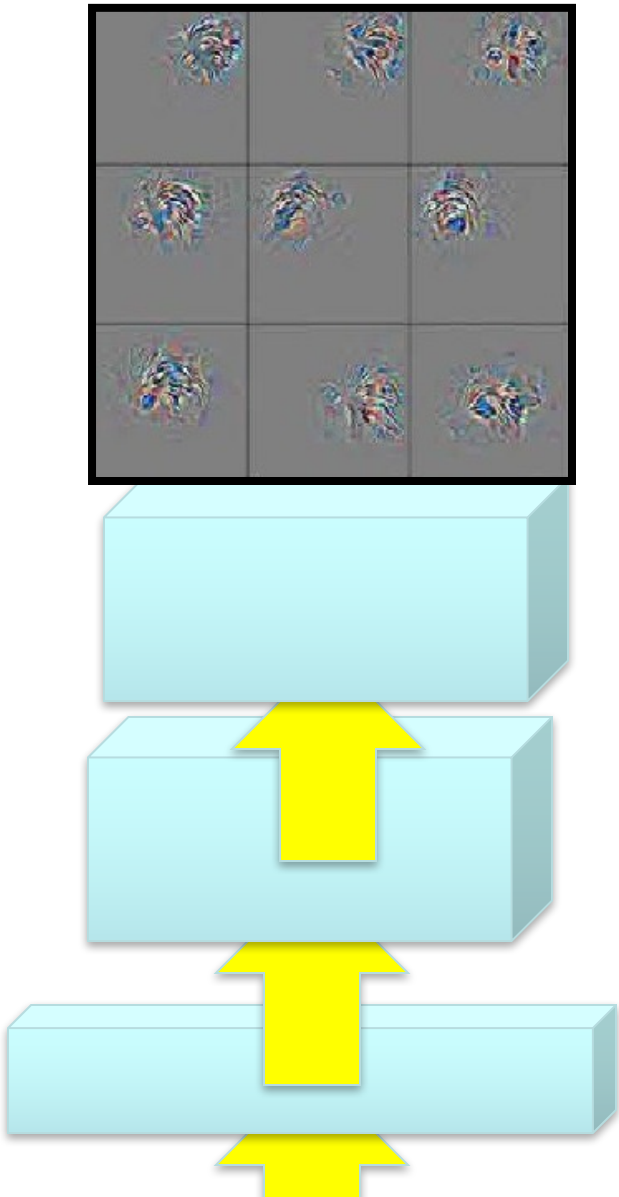
Zeiler and Fergus ECCV 2014

Gather region by region

A spatial pyramid gathers location-dependent evidence

More-layered reduction is better.





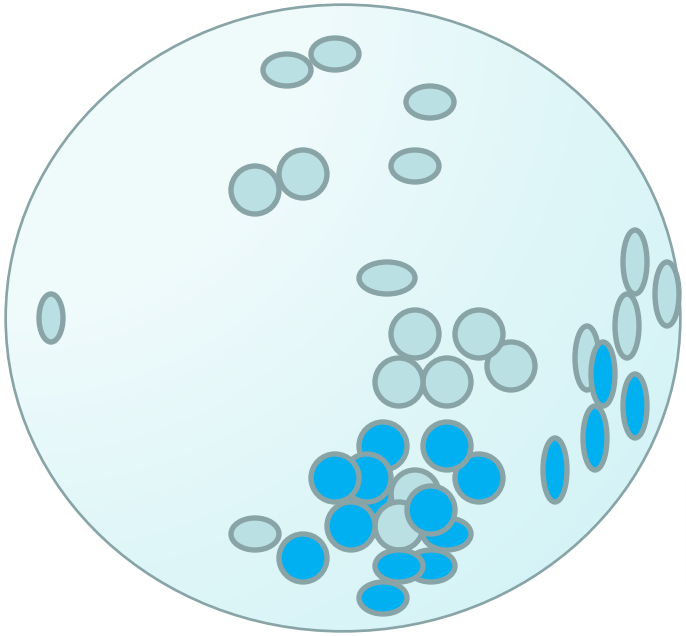
Zeiler and Fergus ECCV 2014

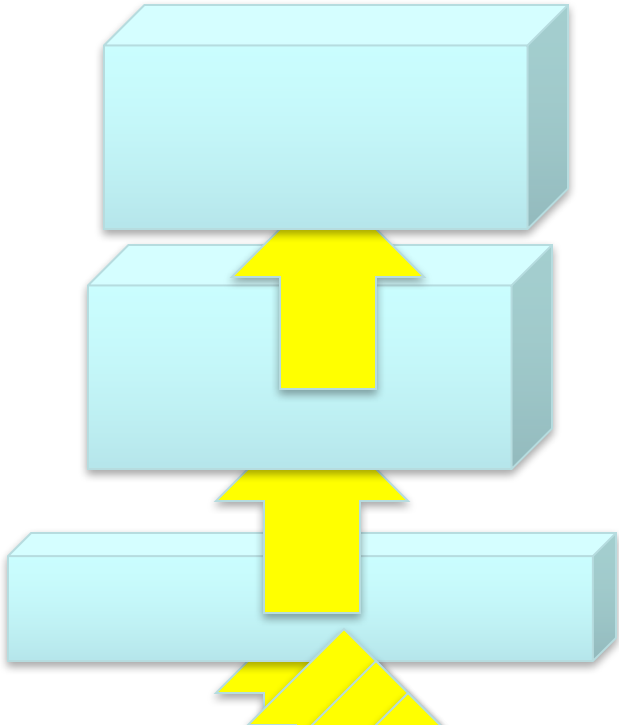
Sample many images

Sample the images in the world:

Learn all relevant variability between types.

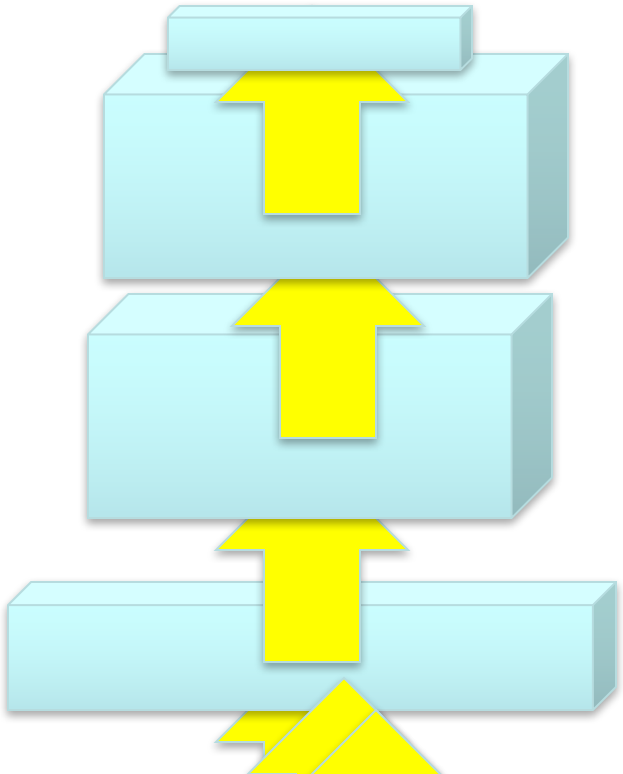
Learn all irrelevant variations not covered by invariance.

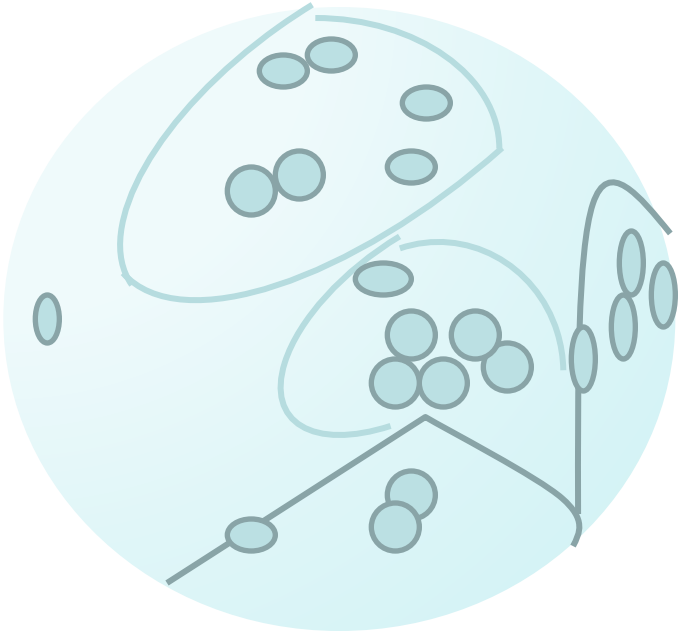




Encode patches into words

Form regions in feature space. More words needed for fine detail discrimination. Words cover several types of patches. Collections of words are still discriminative. Soft assign to code words. Count.



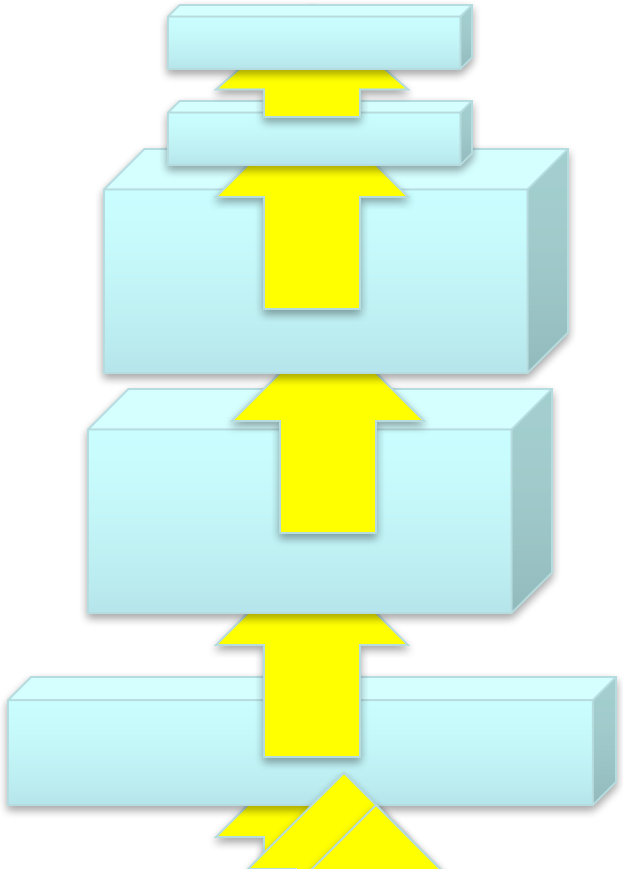


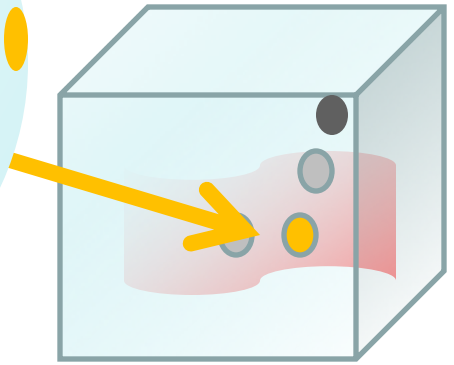
Classify histogram of words

Learn the word-count per type. Classify similarity profile.

Nguyen et al. arXiv 2014

Windsor tie: 0.998959





BoW and CNN's

Features large dimensionality for omni-potency.

powerful normalization & invariance.

Multi-stage encode – classify process to separate:

the variabilities of the world.

from the invariability of the class.

Overview Day 1

1. Introduction, types of concepts, tasks, knowledge as invariance
2. Observables, color, space, time, texture, Gaussian family
3. Invariance, the need for, color invariants, SIFT
4. BoW overview encode and classify
5. Encoders, sparse coding, autoencoders, Fisher vectors
6. SVM, separability, kernels
7. Finale: BoW vs CNN

Tomorrow

1. Vision in the Deep Learning Era I
2. Vision in the Deep Learning Era II
3. Action recognition by learning